

(19) World Intellectual Property Organization  
International Bureau



(43) International Publication Date  
16 August 2001 (16.08.2001)

PCT

(10) International Publication Number  
**WO 01/59406 A1**

(51) International Patent Classification<sup>7</sup>: **G01C 17/38**

(21) International Application Number: **PCT/US01/04277**

(22) International Filing Date: 9 February 2001 (09.02.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/181,706	11 February 2000 (11.02.2000)	US
60/183,283	17 February 2000 (17.02.2000)	US
60/183,374	18 February 2000 (18.02.2000)	US

(71) Applicant (for all designated States except US): **EM-BEDDED LAB TECHNOLOGIES, LLC** [US/US]; 150 Haven Avenue, Port Washington, NY 11050 (US).

(72) Inventors; and

(75) Inventors/Applicants (for US only): **MIN, Kyung, Yang** [KR/US]; 27 Sandy Hollow Road, Port Washington, NY 11050 (US). **PASSE, Paul, J.** [US/US]; 2451 Babcock Road, Hinckley, OH 44233 (US).

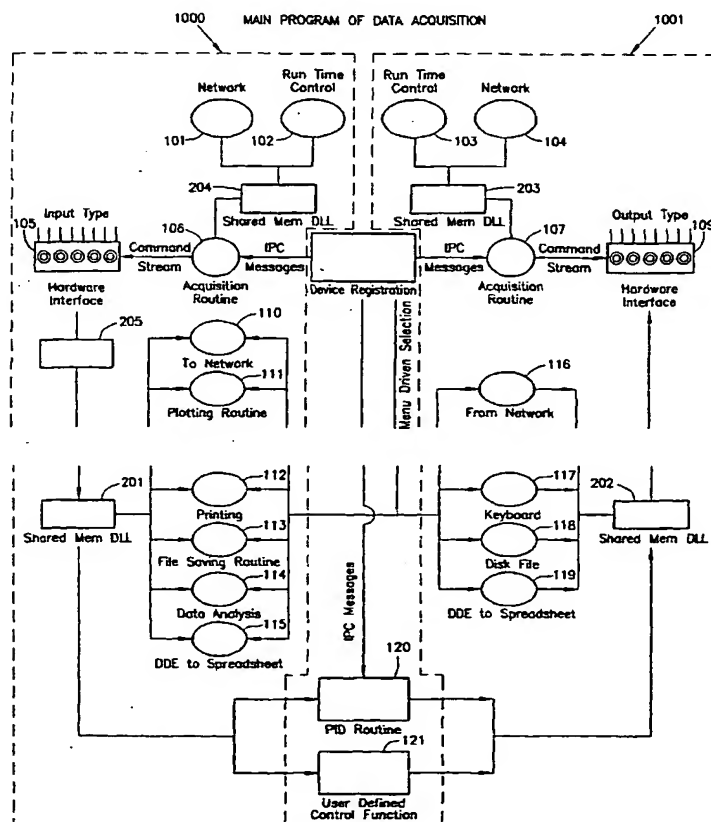
(74) Agent: **KAPPEL, Cary, S.**; Davidson, Davidson & Kappel, LLC, 14th Floor, 485 Seventh Avenue, New York, NY 10018 (US).

(81) Designated States (national): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian

[Continued on next page]

(54) Title: VIRTUAL INSTRUMENTATION SYSTEM AND METHOD



(57) Abstract: A method for controlling at least one instrument (2) is provided which includes the steps of: a) creating a shared memory (201-204); b) selecting an I/O interface (105 and 106) for communicating with an instrument (2); c) inputting into a document, as text, a set of vendor-supplied instructions for said instrument (2); d) inputting into the document, as text, a set of values for run time variables for said instrument (2); e) storing the text of the document in the shared memory (201-204); f) accessing the shared memory (201-204) and translating, with an interpreter, the text of the document into digital signals for communicating with the instrument via the I/O interface (105 and 106); g) controlling the instrument via the translated digital signal. The instrument described herein can be a DMM (4) or other laboratory instrument, or can include an embedded device which can be a module which includes a parallel port (2017), DIO (3017), DAC, ADC, serial port, or GPIB interface for communicating with a downstream device.

WO 01/59406 A1



patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

*For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.*

**Published:**

— with international search report

## VIRTUAL INSTRUMENTATION SYSTEM AND METHOD

This application claims priority from Provisional Application Serial No. 60/183,374, filed February 18, 2000, entitled Virtual Instrumentation System and Method,

- 5 Provisional Application Serial No. 60/183,283 filed February 17, 2000, entitled System and Method for Virtual Instruments, and Provisional Application Serial No. 60/181,706, filed February 11, 2000, entitled Virtual Instrument System and Method, the entire disclosures of which are hereby incorporated by reference.

10 Field of the Invention

The present invention relates to the field of virtual instrumentation, and hardware and software for use with the same.

Background Information

- 15 An instrument is a device which collects data or information from an environment or unit under test and displays this information to a user. An instrument may also perform various data analysis and data processing on acquired data prior to displaying the data to the user. Examples of various types of instruments include oscilloscopes, digital multi-meters, pressure sensors, etc., and the types of information which might be collected by respective instruments include voltage, resistance, distance, velocity, 20 pressure, frequency of oscillation, humidity or temperature, among others.

- Modern instrumentation systems are moving from dedicated stand-alone hardware instruments such as oscilloscopes, digital multi-meters, etc., to a concept referred to as virtual instrumentation. Virtual instrumentation uses general purpose personal computers and workstations combined with instrumentation software and hardware to 25 build a complete instrumentation system. In a virtual instrumentation system, a virtual instrument (VI) operating on a central computer controls the constituent instruments or data acquisition devices from which it acquires data which it analyzes, stores, and presents to a user of the system. Computer control of such instrumentation has become increasingly desirable in view of the increasing complexity and variety of instruments

available for use, and computerized instrumentation systems provide significant performance efficiencies over earlier systems for linking and controlling test instruments.

The two most popular hardware interface options currently available for instrumentation systems are IEEE 488-controlled instruments (GPIB instruments) and RS-232 controlled instruments.

The GPIB (general purpose interface bus) began as a bus designed by Hewlett-Packard in 1965, referred to as the Hewlett-Packard Interface Bus (HPIB), to connect their line of programmable instruments to their computers. The use of this bus was later expanded to communicate with computers manufactured by companies other than Hewlett-Packard and hence the name General Purpose Interface Bus (GPIB) became more widely used than HPIB. The GPIB interface bus was later accepted as IEEE standard 488-1975, and has evolved to ANSI/IEEE standard 488.1-1987. In order to further improve on this standard, two new standards were drafted, these being ANSI/IEEE 488.2-1987 and the SCPI (Standard Commands for Programmable Instruments) standard. The IEEE 488.2 standard purportedly strengthened the original standard by removing ambiguities of the IEEE 488.1 standard by defining data formats, status reporting, a message exchange protocol, IEEE 488.2 controller requirements, and common configuration commands to which all IEEE 488.2 instruments must respond in a precise manner. In 1990, the Standard Commands for Programmable Instruments (SCPI) standards were promulgated, which used the command structures defined in the IEEE 488.2 standard and formed a single, comprehensive programming command set that is used with any SCPI instrument.

A computer can also control an instrumentation system through the computer's serial or RS-232 port. There are currently thousands of instruments with an RS-232 interface. Moreover, computers can also control instruments using the computer's parallel port.

Other interfaces and bus protocols are also used in the art. For example, The VXI

(VME eXtension for Instrumentation) bus is a platform for instrumentation systems that was first introduced in 1987 and was originally designed as an extension of the VME bus standard.

5 The wide variety of possible testing situations and environments, as well as the wide array of instruments available, often make it necessary for a user to develop a program to control respective instruments in the desired instrumentation system. Implementation of such systems generally requires the involvement of a programmer to develop software for data acquisition, analysis and presentation of instrumentation data.

10 Experience and research has shown that current virtual instrument data acquisition solutions are too complex to be readily usable. Most require extensive knowledge in advanced programming and programming techniques, process control, and operating systems as well as a comprehensive knowledge of the instruments to be controlled. This makes the lead time for putting a system into place very long and expensive. It is expensive in terms of development costs, testing, and loss of productivity.

15 In an effort to cut this lead time, most developers focus only on what is absolutely essential in bringing up their system. A direct consequence of this shortsightedness is the inability to reuse the solution for other projects and the subsequently higher maintenance costs. Upgrades, too, are difficult to incorporate, usually requiring extensive rewriting of existing systems. Since these systems were designed for a  
20 dedicated purpose, these changes often introduce flaws that may be difficult to control. Hence, the entire system is compromised. The result is systems that are run at less than optimal efficiency and treated with a "hands off" or "if it ain't broke, don't fix it" mentality. This scenario is generally true in research and industry using the currently available "off the shelf" products like LabVIEW and HP Vee.

25 The inventor of the present invention has recognized that popular products such as LabVIEW and HP Vee adopt a graphical user interface ("GUI") without clear programming methodology and require a detailed knowledge of programming within

their highly specialized environments. Without clear design principles, these products fail to be effective as it often leads to poorly written code which is expensive to maintain. For instance, GUI programs very often spread over a large number of pages and may be several layers of instructions deep. Many times these programs are non-  
5 planar, requiring control lines to cross each other over and over again. As a result, the real-world implementations of these products resemble what is commonly referred to in the industry as "spaghetti code". As a consequence, the use of these products result in programs which are very difficult to modify, maintain and debug. As such, it may take months of training before becoming sufficiently fluent to make even the simplest of  
10 programs and it is very difficult for even the most experienced traditional programmers to implement algorithms of even a moderate size that are verifiably reliable and correct. An alternative is to use one of their prepackaged products (e.g. libraries) to control intended instruments, but this often constrains the user.

#### Summary of the Invention

15 In accordance with the present invention, a virtual instrument system and method is provided which allows the user to make immediate use of his or her instruments without the lengthy process of program development currently needed in commercial products. It is designed to remove the burden of programming from the user and allow for nearly immediate use of instruments without learning a new programming language  
20 and a minimal amount of coding.

In accordance with a first embodiment of the present invention, a computerized method for controlling at least one instrument coupled to a computer is provided, comprising the steps of selecting an I/O interface for communicating with an instrument; inputting into a document, as text, a set of vendor-supplied instructions for said instrument;  
25 inputting into the document, as text, a set of values for run time variables for said instrument; translating the text in the document into digital signals for communicating with the instrument via the I/O interface; and controlling the instrument via the translated digital signals, wherein said step of controlling includes one or more of the steps of transmitting commands to said instrument and receiving data from said

instrument and transmitting commands and data to said instrument. In accordance with the present invention, the steps of inputting a set of values for run-time variables, translating the text, and controlling the instrument can be performed sequentially or concurrently. In this manner, it is possible to alter the values of the run-time parameters  
5 "on the fly" during the execution of an experiment or test without stopping and restarting program execution. Preferably, the method is implemented using shared memory for storing at least the set of values for run-time variables. The method can be used to control both input-data-type instruments such as digital multi-meters and output-data-type instruments such as D/A converters.

10 In accordance with a second embodiment of the present invention, a method for controlling at least one instrument is provided which includes the steps of : a) creating a shared memory; b) selecting an I/O interface for communicating with an instrument; c) inputting into a document, as text, a set of vendor-supplied instructions for said instrument; d) inputting into the document, as text, a set of values for run time  
15 variables for said instrument; e) storing the text of the document in the shared memory; f) accessing the shared memory and translating, with an interpreter, the text of the document into digital signals for communicating with the instrument via the I/O interface; g) controlling the instrument via the translated digital signals.

For a input-data-type instrument, step (g) includes transmitting commands to said  
20 instrument and receiving data from said instrument, and the method proceeds to: h) store the data receiving from the instrument in a second shared memory, i) access the second shared memory, and process the data contained therein with a service routine; wherein at least steps c through i can be performed sequentially or concurrently.

For a output-data-type instrument, step (g) includes transmitting commands and data to  
25 said instrument, and accessing a second shared memory to obtain data to transmit to said instrument; and the method proceeds to: h) write data to be transmitted to the instrument to the second shared memory with a service routine; wherein at least steps c through h can be performed sequentially or concurrently.

Preferably, the method in accordance with the present invention is implemented using an event driven architecture such as IPC. The use of such an architecture improves reliability by reducing the likelihood that an run-time error, for example in the interpreting process will cause the rest of the application to crash.

- 5 In accordance with further aspects of the first and second embodiments, a computerized method is provided for controlling a plurality of instruments, and a plurality of documents are generated, each document including, as text, a set of vendor-supplied instructions for said instrument and a set of values for run time variables for said instrument. In accordance with this embodiment, a respective interpreter is provided  
10 for each instrument, and each interpreter executes as a separate thread in an event driven architecture. The use of this architecture will, for example, reduce the likelihood that an run-time error from one instrument will affect processing of the remaining instruments.

- In accordance with other aspects of the first and second embodiments, the step of  
15 inputting a set of run time variables further includes the steps of creating, with a graphical user interface, a run-time window, the run-time window including input fields for entering each of the values for the run-time variables and writing said set of values for the run-time variables into said document.

- In accordance with other aspects of the first and second embodiments, the instrument  
20 comprises at least one embedded device and at least one downstream device, the at least one downstream device being coupled to the I/O interface via the embedded device. The embedded device communicates with the I/O interface via a LAN or WAN, and most preferably communicates with the I/O interface via an Internet protocol. In this regard, both the I/O interface and the embedded device preferably include network  
25 interface cards to support this communication, and most preferably, the embedded device is programmed to communicate using TCP/IP protocol. The embedded devices may be coupled directly to the I/O interface via the LAN or WAN, or, alternatively, may be routed through an intermediate controller.

Brief Description of the Drawings

Figure 1 shows an illustrative virtual instrument system in accordance with an embodiment of the present invention including a DAQ system, an off-the-shelf  
5 instrument, and an embedded instrument.

Figure 2 shows a flow chart for a main program of the DAQ system of Figure 1.

Figure 3 shows a main graphical user interface (GUI) for the main program of Figure 2.

Figure 4 shows a GUI for a run-time control window for providing the run-time controls of Figure 2.

10 Figure 5 shows a GUI for a plotting routine of Figure 2.

Figure 6 shows a GUI for a PID routine of Figure 2.

Figure 7 shows a GUI for a Data Analysis routine of Figure 2.

Figure 8 shows a parallel port embedded device module in accordance with a preferred embodiment of the present invention.

15 Figure 9 shows a DIO embedded device module in accordance with a preferred embodiment of the present invention.

Figure 10 shows a DAC embedded device module in accordance with a preferred embodiment of the present invention.

20 Figure 11 shows an ADC embedded device module in accordance with a preferred embodiment of the present invention.

Figure 12 shows a Serial Port embedded device module in accordance with a preferred embodiment of the present invention.

Figure 13 shows a GPIB embedded device module in accordance with a preferred embodiment of the present invention.

- 5     Figure 14(a) shows an embedded device controller in accordance with a preferred embodiment of the present invention, coupled to a plurality of embedded devices.

Figure 14(b) illustrates the relationship between various application programs residing on the embedded device controller of Figure 14(a).

- 10     Figure 15 shows an illustrative system providing client computers with remote access to a plurality of devices via a DAQ computer.

Figure 16 shows an illustrative system providing client computers with remote access to a plurality of devices via an intermediate controller.

#### Detailed Description of the Preferred Embodiments

- 15     The preferred embodiments of the present invention will now be described in detail with reference to Figures 1 through 16. Although the system and method of the present invention will be described in connection with these preferred embodiments and drawings, it is not intended to be limited to the specific form set forth herein, but on the contrary, it is intended to cover such alternatives, modifications, and equivalents, as can  
20     be reasonably included within the spirit and scope of the invention as defined by the appended claims.

#### Overview

- 25     The present invention provides a reliable, robust, versatile, yet simple solution to data acquisition and process control. Event driven programming makes the DAQ system in accordance with the preferred embodiments of the invention easy to install and run.

The system provided allows the user to concentrate on commanding and controlling devices rather than spending time on authoring narrowly focused complex programs.

In accordance with the present invention, a DAQ system is provided which is comprised of a "software suite" of several independent programs (or processes) running concurrently on a computer in a multitasking environment, with each process controlling a specific aspect of the data acquisition and analysis. Preferably, many of these programs can be run in stand-alone mode for ease of development and testing. Coordination of the concurrently running programs is achieved through a combination of shared memory, FIFOs (First-In First-Out) and IPC (Inter-Process Communication), which are overseen by the user through GUI (Graphical User Interface).

The user interface is a unique paradigm in DAQ (data acquisition) and analysis. It allows the operator to automatically start up hardware, initialize all the various processes of the system, easily navigate the states of the online system, and monitor the progress of the runs. Comprehensive graphical display of the user interface identifies all aspects of DAQ that a researcher typically employs. The GUI puts in easy reach (typically one click) data analysis, device commanding, data logging, data visualization, and process feedback controls.

Built-in analysis features provides a spreadsheet-like interface and include multi-axis plots of data traces, comprehensive uni-variate and multi-variate statistical functions, advanced mathematical functions, and signal processing.

The exemplified software suite can run on any IBM-compatible PC's with 486 CPU or better under MICROSOFT® WINDOWS® 95, 98, 2000 and NT operating system, and can be written in a visual programming environment (using such object-oriented languages as Delphi and C++ Builder). It should be noted, however, that systems in accordance with the present invention can also be constructed for use with other operating systems, such as UNIX, SOLARIS, LINUX, MAC OS, etc., and using other programming languages.

### 1. Macroscopic vs. Microscopic Approach

The preferred system in accordance with the present invention has been carefully engineered to address most of the above-mentioned deficiencies in the prior art. A philosophical decision was made to implement in the software suite an architecture  
5 which is very similar architecture which is found in the hardware of an IBM-compatible PC.

From the view point of customers, the basic building blocks of such PC hardware are a general-purpose motherboard and a few plug-in cards useful to specific applications. On the other hand, individual chips and wires connecting those chips are the building  
10 blocks from the view point of manufacturers. While manufacturers inevitably take a microscopic approach to build the motherboard and/or a plug-in card by assembling chips and wiring them, customers take a different approach when they purchase and customize their own computer units. What matters to a customer is the task (or function) assigned to each card. Structural details of each card are ignored. A customer  
15 simply buys any card which is capable of performing a specific task (e.g., hard disk controller, sound or modem cards), plugs it into any empty slot of the motherboard, and it works immediately.

One of the advantages of IBM architecture lies in the fact that a customer can take such a macroscopic approach. One of IBM's major contributions was to organize the basic  
20 computer concepts, and quality control well known in the world of their bigger machines and apply them to the microcomputer. IBM successfully introduced a hardware architecture whose basic building blocks are not microscopic chips but, rather, macroscopic tasks (or cards).

There is a fundamental difference between these two approaches. Using a microscopic  
25 approach, an individual should buy many chips and wire them carefully to build a card. Each chip has specific input and output pins which can be understood only after one

carefully reads through each chip's data sheet. It may take several days for a reasonably skilled expert or hobbyist to build even a simple card. This approach requires a hardware creation process which is complex, lengthy and expensive. Moreover it is very difficult to debug if any human error occurs. However a macroscopic approach is  
 5 free from such a hardware creation process. As a result, it is simple, fast and economical. It does not require any debugging process.

When applied to a construction of a DAQ software suite, the choice of building blocks results in a similar difference in resulting structure. In a microscopic approach, the following correspondence (or similarity) between hardware and software architecture  
 10 can be seen:

Chip	Object
Wiring	Line Drawing

In contrast, a macroscopic approach results in the following correspondence (or similarity) between hardware and software architecture:

15	Motherboard	Structure
	CPU	Main Program
	Memory	Shared Memory
	Clock & Interrupt	IPC Message
	Keyboard	Script File
20	Plug-in Card	Task (Function)
	Hardware Interface	User Interface

It is this similarity which motivated development of the DAQ software suite in accordance with the present invention. In the following it will be shown that the resulting software provides many advantages over existing DAQ software suites such as  
 25 LabVIEW and HP Vee.

#### i) The Microscopic Approach as Used in the Prior Art DAQ Systems

In the graphical programming solutions used in LabVIEW or HP Vee, the primitive building blocks are objects which create, analyze, or display the data. A diagram is created using objects and lines during program creation process. The lines primarily represent data and execution flow typically follows the data flow. All the objects have  
5 input and output pins. These pins are used to create subroutines (User Functions or VI's) by connecting to pins of other objects through lines. Therefore, lines representing data can either be inputs to the object or outputs from the object. These pins are also used to sequence data or execution flow.

This approach is very similar to the one adopted in circuit design. Unfortunately, too  
10 few programmers have the experience with the circuit design techniques within this approach. The followings are some of the most common difficulties encountered by user's while using such an architecture:

- The programmer has to select proper objects from among numerous objects given in the pull-down or pop-up menus.
- 15 • In many cases, the objects utilized would require hundreds of lines to be drawn. Often times, many of the lines must cross over each other repeatedly. As a result, a multi-layer window scheme is often required.
- It is very difficult, even for the author, to derive the execution  
20 scheme from the interface. The finished diagram can not be easily interpreted by either the author or other programmers.

As a result, the program creation process is extremely complex.

ii) the Macroscopic Approach in Accordance with the Present Invention

In accordance with the present invention, a system is provided which offers very

different solutions to graphical programming. The primitive building blocks are structure and tasks. Tasks are properly arranged within the structure so that data flow and execution sequence are automatic. The following software design principles were employed in order to decide the structure, the nature of each task and the arrangement of all the tasks within the structure.

#### General Programming Principles

1. The software should not require any compiler and should be fully open to the user at any level. However it should have all of the capabilities of any full-featured compiler.
2. The software should provide high level graphics in addition to math, instrument control, general purpose I/O, and data acquisition capability.
3. The programming should be event driven. Sequence, looping and flow of control are natural features of all the modern event driven languages.

#### Structure-related principles

1. The structure should be simple enough to impose virtually no programming work on the user. It should be intuitive enough to allow the user to ascertain its state at a glance. It should be well-organized enough to allow an automatic data flow and execution sequence.
2. The structure should have an input for each input-data-type instrument (such as A/D converter or DMM) and an output for each output-data-type instrument (such as D/A converter). In addition, the structure should provide a bridge between input-data-type and output-data-type instruments. The bridge is useful

for programming, for example, user functions such as PID (Proportional, Integral and Derivative) routines.

- 5 3. The structure should allow the user to control any type of instrument. Therefore tight coupling of instrument-specific commands with the remaining program should be avoided. In this regard, it is preferable that the instrument's commands not be "hard coded" into the controlling software.
- 10 4. The structure should maintain shared memories to provide the ability to share data among different tasks. Such memories should be implemented via DLL's (Dynamic Link Libraries).
- 15 5. Since tasks need to communicate with each other to synchronize all the concurrently running tasks or to event/time trigger other tasks, the structure should have IPC.

#### Task-related Principles

- 20 1. There should be a sufficient, but minimal number of tasks, in order to economically include all the possible laboratory-related human activities.
2. All the tasks should have visual representation. The user interface of the task must be familiar to the customer, using natural metaphors.
- 25 3. Special tasks should be provided which allow remote monitoring and controlling. As a result, one can remotely access the system to either query the state of experimental runs or to perform any commanding of instruments.

4. At least one special task should be provided which can dynamically set some parameters for the preselected instructions.
5. There should be one special task performing the role of interpreter, i.e., a script language to include the ability to evaluate conditional statements, case statement, and loops.

5

The implementation of the above-referenced software design principles will now be explained in the context of actors, process, interprocess communication, process manager, and services.

10 ACTORS (instruments) are scripted by users with their configuration, control, and termination sequences of instructions. In scripting the actors, the user will employ those commands commonly in use for generally available interface cards (e.g. GPIB) and those specific to his or her device in order to command the behavior of the equipment. The user will enter these instructions via a common spreadsheet-like user interface which facilitates understanding, reliability, correctness, and ease of use.

15 PROCESSES are the runtime instances of the actors having the responsibility of either collecting data from or issuing commands to the associated instrument.

INTERPROCESS COMMUNICATIONS (IPC) is the means by which an actor is signaled to perform its designated role within the data acquisition context defined by the user and the vehicle by which it shares the related data.

20 PROCESS MANAGER is a process responsible for the simultaneously managing and providing the timing of information, availability of information, signaling of processes, and runtime services for the user.

SERVICES are those tasks that provide for the data analysis tools, data visualization

(plotting), logging, enabling/disabling of instruments, and overall control of the current data acquisition run.

Figure 1 shows a DAQ system 1 in accordance with a preferred embodiment of the present invention. Figure 2 is a flow chart for a main program 100 for the DAQ system 1, which shows the schematics of structure and tasks as well as interprocess communication among processes and data stream.

Referring to Figure 2, the DAQ system 1 includes an interface 105 for communicating with input-data type instruments (such as a digital multi-meter) and an interface 109 for communicating with output-data type instruments (such as a D/A converter). Data acquisition and processing for the input-data type instruments is controlled via component set 1000 and data acquisition and processing for the output-data-type instruments is controlled via component set 1001, with components 107, 120, and 121 serving both input and output data type instruments are described below. Although the invention is described herein as providing with ability to control both input-data-type and output-data-type instruments, it should be apparent that a system could be constructed to control only input-data-type instruments (by employing only component 107 and component set 1000) or only output-data-type instruments (by employing only component 107 and component set 1001).

Referring to Figure 2, four shared memories 201-204 are implemented via DLL'S, and permit several WINDOWS® applications to share data and provide all the services described below. Moreover, the shared memories may also be used to bridge the input and output-data-type by providing a temporary buffered layer. In this regard, Figure 3 shows a PID control window, where the PID routine 120 of Figure 2 bridges two shared memories 201 and 202, allowing data received from an input-data-type instrument to be used to control an output data-type instrument.

The Main Program 100 of Fig. 2 performs six major functions.

2. Register instruments by user-defined device and channel names, together

with types of the hardware interface which the user wants to use (e.g., GPIB, RS232 or parallel port). The Device Registration component 107 in Fig. 1 works as a process manager, maintaining timing and synchronization of all the events, in addition to device registration.

- 5           3. Provides a spreadsheet-like interface 302 (See Figure 3) in which the user can type in vendor-supplied instrument-specific instructions including commands and run-time values during the initial design phase. The run-time values are automatically copied into one of two shared memories (Shared Mem DLL 203 or 204) and can be accessed by any  
10           client program (Acquisition Routines 106 or 108 of Fig. 2). Drag-and-drop operations of graphical icons from the menu bar 301 into the spreadsheet 302 allows the user to construct a runtime-control window for controlling the run time values by collecting and rearranging such icons in a separate window. Figure 4 shows such a runtime window.  
15           These runtime controls are associated with the actors (instruments) by one or more of shared memories 201 (for controlling processing of data received from input-data-type instruments), 202 (for controlling processing of data to be sent to output-data-type instruments), 203 (for controlling run-time values to input-data-type instruments), and 204 (for  
20           controlling run-time values to input-data-type instruments).
4. Create and maintain four shared memories 201-204 which can be used to provide the ability to share data among different processes. For processing data received from an input-data-type instrument, the user can access data residing in shared memory 201 in order to get services  
25           110-115 such as plotting 111, printing 112, file saving 113, data analysis 114, DDE to Spreadsheet 115, and To Network 110. As an example, Figure 5 shows a chart where three signals are plotted with three different Y-scales using the plotting routine 111. For processing data to be transmitted to an output-data-type instrument, the user can access

data residing in shared memory 202 in conjunction with services From Network 116, Keyboard 117, Disk File 118, and DDE to Spreadsheet 119. This architecture not only simplifies data access, but also prevents the user from having to code various file-buffering schemes. All the service routines in Fig. 2 are replaceable by user-supplied routines.

5

5. Maintain standard service routines (e.g. plotting 111, printing 113 and file saving 113) and register user-supplied service routines (e.g. data analysis 114, network 110 and DDE 115 (Dynamic Data Exchange)) in the menu area.

10

6. Define and register user-supplied IPC messages. Object-oriented programs are very chatty by nature, frequently sending messages between different objects. The Main Program 100 monitors timing and synchronization of such event/timer triggered windows messages. All the service routines can inherit (or share in) the currently active IPC messages. The IPC messages in Fig. 2 represent event/timer triggered windows messages. The user-supplied service routines (e.g. 114 and 115) can easily share in the currently active IPC message system by registering such messages in the routine using standard Windows API functions.

15

20

7. Each client, which is a separate executable program with a separate thread, and which is designated as Acquisition Routines 106 (for retrieving data from input-data-type instruments) and Acquisition Routines 107 (for sending data to output-data-type instruments), shares instrument specific instructions with the Main Program 100 via shared memories 203 and 204 respectively and issues commands to the associated instrument by translating these instructions into machine language. It maintains a software handshake with the 'Main program' by exchanging IPC messages. The data acquisition routines 106 and 107

25

automatically interpret the commands typed in the spreadsheet 301 in view of the run-time control values stored in shared memory and translate them into machine language. Each process in this software will automatically acquire a different thread.

5 In accordance with the preferred embodiment of the present invention described above, a user can create and run an experiment via virtual instrumentation by simply

1. registering the desired instruments and selecting the type of hardware interface to be used to communicate with each instrument;
2. typing in a set of vendor-supplied instructions in a spreadsheet,
- 10 3. rearranging some graphical icons to form a runtime control window,
4. selecting a proper client (106 or 108) based on the hardware interface chosen and simply run it, and
5. running service routines by mouse-clicking the corresponding menus.

Everything else is automatically taken care of by the software. As a result, it takes a  
15 relatively short a time to program sophisticated instruments. For example, a digital multi-meter (DMM) can be configured and run in less than 4 hours.

The manner in which the DAQ system of Figures 1-7 operates will now be discussed in detail. Figure 3 shows the graphical user interface 300 (GUI) for the main program of Figure 2. Field 305 lists five devices (e.g. instruments) as currently registered in the  
20 DAQ system: input-data-type devices Dev1 through Dev3, and output-data-type devices Dac1 and Dac2. Field 306 allows a user to monitor data transfers through hardware interfaces 105 and 109. When the switch in field 306 is in the "on" position, the receiving data field will flash when data is transferred through interface 105 and the transmitting data field will flash when data is transferred through interface 109.

25 Spreadsheet-like interface 302 is shown in Figure 3 for Dev1. The corresponding interfaces 302 for Dev2, Dev3, Dac1, and Dac2 are hidden from view, but are accessible by "clicking" on tabs 311-314. In Figure 3, the interface 302 for Dev1

contains command lines 1-17 which are used to control a Digital Multi-meter (DMM) via a GPIB communications protocol supported by the DMM. As one of ordinary skill in the art will appreciate, user manuals for such DMM's include detailed instructions regarding the specific command protocols which must be used to communicate with the device via each particular supported interface protocol (e.g. GPIB, RS232). In order to initialize Dev1, the user simply types the appropriate commands into the spreadsheet-like interface 302.

In the interface 302, column 315 provides the line number and column 316 indicates if the instruction at that line number is an internal information instruction (i), a device setup instruction (s), a device write instruction (w), or a device read instruction (r). Column 317 indicates the type of GPIB instruction listed, wherein "info" indicates a post-processing command which is executed after the data is received from the device; ibwrt indicates a GPIB write command, and ibrd indicates a GPIB read command. Column 318 includes the manufacturer-designated GPIB commands for controlling the DMM. In this regard, variable parameters in the command are surrounded by "\$". Column 319 includes the variable parameter field for the instructions (e.g. \$x0\$ through \$x8\$), and column 320 includes an initial value for each variable parameter.

In the example shown, the DMM has been set to provide data on channels 1 through 5 in scanner mode (line 110), with three channels reading resistance (line 5), and two channels reading voltage (line 8). Variable control parameters are provided for selecting which channels will read resistance and which channels will read voltage. Additional variable control parameters are provided in lines 6 (NPLC), 7 (Average), 9 (NPLC), and 10 (Average) for other DMM specific instructions. Initial values for the control parameters appear in column 320, with, for example, DMM channels 1, 2, and 3 initially set to read resistance and DMM channels 4 and 5 initially set to read DC voltage.

Columns 321 and 322 are used to provide run-time controls for the DMM. These run-time controls can be varied at any time, and allow the user to reconfigure the DMM "on

- the fly” without re-starting or otherwise stopping execution of the program. Column 321 provides a label for the run-time control. In order insert a label, the user “clicks” on the label button 334 in menu bar 301, and then types in the desired label. In Figure 3, the label “AveNo” has been inserted at line 1 to indicate that the value in column 322 is the number of data values that the program will average from each DMM channel; the label “channels” has been inserted at line 5 to indicate that the values in column 322 list the DMM channels which will read resistance, the label NPLC has been inserted at line 6 to indicate that the value in column 322 is the NPLC value for the resistance measurements; the label “average” has been inserted at line 7 to indicate that, for resistance measurements, the DMM will transmit the average of three resistance measurements to the interface 105; the label “channels” has been inserted at line 8 to indicate that the values in column 322 list the DMM channels which will read DC voltage, and the label NPLC has been inserted at line 9 to indicate that the value in column 322 is the NPLC value for the voltage measurements.
- 15 In order to insert a run-time variable field into column 322, the user “clicks” on one of the buttons 323-333, 335, 336 in menu bar 301. In this regard, for example, button 328 allows the selection of an integer on a sliding scale (as shown in line 7), button 335 allows the listing of a combination of elements (as shown in lines 5 and 8), button 331 allows the listing as a “spinning” integer (as shown in line 1), and button 332 allows the listing of a “spinning” fraction (as shown in lines 6 and 9). In the example shown, it should be noted that the run-time values have been altered from the initial values, and that now, for example, DMM channels 1, 4, and 5 are reading resistance, while DMM channels 2 and 3 are reading DC voltage.

- 25 The labels and run-time values of columns 321 and 322 can also be moved into a “run-time window” as illustrated in Figure 4. In order to create a run-time window, the user “clicks” on the edit button 337, selects “duplicate” (not shown), and the labels and run-time values of columns 321 and 322 are copied into the run-time window of Figure 4, where the user can arrange them on the screen as desired. In Figure 4, the Dev1 labels and run-time values are arranged in the center column, with the labels and run-time

values for Dev2, Dev3, Dac1, and Dac2 shown in the side columns. It should also be noted that service routines, such as routines 110 through 121 may also be copied into the run-time window of Figure 4.

To begin an experiment, a user clicks on the start button 350. Referring again to  
5 Figures 1 and 2, the run-time values from the spreadsheet-like interface 302 are copied to shared Mem DLL 204. Any run-time changes to the run-time values are immediately copied to Shared Mem 204 so that the values in shared memory are always up to date. Acquisition routine 106 (which can be a script interpreter or a compiled routine) reads  
10 the commands 318 from the interface 302, and the run-time values from the Shared Mem 204, and generates a command stream to the hardware interface 105, thereby controlling the DMM and reading the requested voltage and resistance data in accordance with the designated run-time values. The data received from the DMM through the interface 104 is then stored in Shared Mem DLL 201 for further processing.

In one embodiment, a filter script filters the data received from the interface 105 so  
15 that it is stored in the Shared Mem DLL 201 in usable format. For example, in many cases, temperature data received from a DMM is received in Ohms. The filter script program could be used to convert the data from Ohms to °C. In addition, the filter script program could be used to provide other internal processing. For example, line 1 of spread-sheet like interface 302 contained an instruction to take the average of every  
20 three values received from the DMM as the actual value. This type of function is useful, for example, for synchronizing the values for devices with different sampling speeds. For example, referring to Figure 2, if Dev1 is three times as fast as Dev2, then the devices can be synchronized by setting the run-time control value in line 1 of spread-sheet like interface 302 to 3. The filter script program 205 could be used to  
25 implement this function. Alternatively, a compiled data filter, as described below, could be used.

The values in Shared Memory 201 are accessible by service routines 110 through 121 as described above. Figure 5 shows an illustrative plotting routine 111 which has been

configured to display channels 1, 2, and 4 as a function of time, with the values of the channels plotted on three separate Y axis 401, 402, 403. Figure 6 shows an illustrative proportional, integral, derivative (PID) routine 120 configured to receive data from channel 4. The output of the PID routine 120 is then available for use as output data for an output-data-type instrument such as a D/A converter. Figure 7 shows an illustrative Data Analysis routine 114, which tabulates the data received in the form of a spread sheet and allows the user to configure the data into graphical form as desired. Referring to Figure 2, the plotting routing 111 can be accessed via pull down menu 360, the data analysis routine 114 can be accessed via pull down menu 362, and the PID routine 120 can be accessed via pull down menu 361.

The implementation for the output-data-type instruments such as Dac1 and Dac2 (Digital to Analog Converters) and for Dev2 and Dev3 in Figure 3 is similar to the implementation for Dev1 described above. Each of these devices is controlled by a respective acquisition routine 106 (in the case of Dev2 and Dev3) or 108 (in the case of Dac1 and Dac2) which runs on a separate thread. Communication is synchronized by the main program using an event driven technology, which, in the illustrated embodiment is IPC. However, in the case of output-data-type instruments Dac1 and Dac2, the available service routines are routines 116 through 121.

#### Compiled Data Filter

As mentioned above, as data is streaming from the connected input devices 2 and instruments 4 into the DAQ system 1, it is captured at interface 105 and stored in a shared memory DLL 201. At the point of capture, however, that data may not be in a format which is directly usable by the system 1. Therefore, in certain embodiments of the present invention, a filter could be provided which converts the raw data stored in the shared memory DLL 201 into a format which is compatible with the system 1. Therefore, in certain embodiments, the software of the system 1 includes an option which allows the user to load a software filter of his creation for use by system software. Preferably, the software filter is in the form of a DLL, and the system supports the creation of such filter software in, for example, C++ or Pascal. IPC signals

the arrival of data in the shared memory DLL and the data filter retrieves the data from the shared memory DLL and sends it to routines 110-115, 120, 121. To support the creation of software filters in these languages, corresponding Win32 capable Integrated Development environments that use the gcc and fpc free-ware compilers can be  
5 provided. Software templates and examples for creating a custom filter can also be provided to assist a user in creating a software filter in the form of a filter DLL.

Preferably, the system also includes an option which allows a user to simply specify the incoming data's format and any conversion formula that should be used on the data. Based upon this information, the system can automatically generate a new filter DLL  
10 and insert the filter between the shared memory DLL and the remainder of the system software described with regard to Figure 2.

The system can also include a library of filter DLLs for a set of popular instruments, and provide a function automatically detects the instrument 2 that is being used, and selects the appropriate filter DLL from the library.

#### 15 Embedded Device Technology

Referring again to Figure 1, in accordance with further embodiments in accordance with the present invention, the actors (or instruments) may include embedded devices 2 as well as currently-available off-the-shelf instruments 4 (such as DMMs). In accordance with this embodiment, a device 2 includes various I/O ports 3 for receiving  
20 unprocessed information such as voltage, resistance, pressure, strain, flow, TC, or digital data stream from a downstream source such as a thermometer, resistor, communications receiver, etc. Embedded in the memory of the device are associated device drivers for receiving the information from each input and converting it to digital form, an operating system configured to support an IP protocol such as TCP/IP or  
25 UDP/IP, an Ethernet adapter, and dedicated software which transmits the digitized input through IP protocol to the DAQ 1 in response to commands from the DAQ system 1. The embedded device may function as either or both of an input-data-type instrument or an output-data-type instrument. Referring to Figure 2, in the context of a

system for controlling embedded devices 2, the interfaces 105 and 109 reside in the embedded devices, and are controlled remotely from the DAQ 1 using, for example, TCP/IP. In this context, the I/O interface of the DAQ 1 would simply be a port for remotely accessing the embedded devices 2, such as a Network Interface Card.

- 5 Figures 8 through 13 illustrate, respectively, six illustrative embedded devices 2: a parallel port module 2000, a DIO module 3000, a DAC module 4000, an ADC module 5000, a serial port module 6000, and a GPIB module 7000. When addressing one of these modules, the user enters the commands in the appropriate protocol 317 (e.g., GPIB, DIO, parallel port, ADC, DAC, and serial port) for the device in the spread sheet  
10 302. As explained in more detail below, these embedded device modules may be connected directly to the computer 1, or may be connected to an intermediate controller 8000, which, in turn, is connected to the computer 1.

- Figures 8 though 13 show embedded modules 2000 through 7000 which communicate with the computer 1 over a LAN and/or WAN (such as the Internet) via the ethernet  
15 protocol 2001, using TCP/IP. LAN connections can, for example, be implemented using conventional 10BASE-T or 100BASE-T coaxial cable. Although the module 2000 will be described below with reference to the ethernet protocol, it should be understood that alternative protocols can also be used, including for example, Token Ring, FDDI, ATM, SONET, and X.25 protocols. Similarly, although the embedded  
20 modules will be described as communicating via Internet protocols such as TCP/IP and UDP/IP, other protocols could similarly be used, including, for example, IBM's SNA, Windows' NetBIOS, Apple's AppleTalk, Novell's NetWare, and Digital's DECnet. Similarly, wireless communications links and associated protocols may also be used.

- The modules 2000 through 7000 include a conventional NIC 2002 ("Network Interface  
25 Card") for receiving and transmitting data via ethernet 2001. Referring to Figures 8-13, the modules 2000 though 7000 includes a crystal oscillator 2006, CPU 2005, interrupt control 2004, brown-out detector 2020, watch dog circuit 2021, flash memory 2008,

SPI 2009, EEPROM (2010 through 7010), RAM 2011, timer circuitry 2022, UART 2012, bus controller 2013, latch 2014, and I/O port 2023. These components are interconnected in a conventional manner to allow commands from DAQ 1 received at the NIC 2002 via the ethernet to be converted into the appropriate control and data signals which are transmitted to the downstream device, and to allow control and data signals from the downstream device to be converted into data to be transmitted to the computer 1 via the NIC 2002 and ethernet 2001. Preferably, the module 2000 includes two separate ROMs, shown in Figure 8 as Flash memory 2008 and EEPROM (2010 through 7010), so that the programming which is generic to all modules 2000 through 7000 is contained in one ROM (preferably Flash memory 2008), whereas the module specific programming is contained in a separate ROM, in this case EEPROMs 2010 through 7010 contains the module specific programming necessary to control the parallel port 2017, DIO 3017, DAC 4017, ADC 5017, UART 6017, and GPIB 7017 communication chips, respectively. Most preferably, the modules 2000 through 7000 are fabricated in a two part construction, with a first circuit board containing only components which are generic to all of the modules 2000 through 7000, and a second circuit board containing the module specific components and optionally some generic components as well. It should be noted that although the components 2002-2023 are illustrated as separate components, two or more of these components may be fabricated in a single integrated circuit chip. For example, in a particularly preferred embodiment of the present invention, an Amtel ATMega103 microcontroller provides the oscillator 2006, CPU 2005, watchdog 2021, flash memory 2008, UART 2012, latch 2014, timer 2022, and interrupt control 2004 of modules 2000-7000.

Figure 8 illustrates a parallel port module 2000. Data transmitted from the computer 1 via the ethernet 2001 and TCP/IP protocol is received at the NIC 2002, and is converted via CPU 2005 into the proper parallel port control and data signals, which are passed to parallel port 2017 through I/O ports 2023, and transmitted to a downstream device. Similarly, control and data signals received from a downstream device, are received at parallel port 2017, pass through I/O ports 2023, and are converted via CPU 2002 into the proper TCP/IP ethernet protocol, and transmitted to the computer 1 via NIC 2002

and ethernet 2001. These conversions are implemented in a conventional manner via components 2004 through 2023, and therefore, will not be discussed in detail herein. Exemplary downstream devices which can be controlled via the module 2000 include digital multimeters, printers, scanners, CD Rom drives, ZIP drives, and CNC machines.

5 As one of ordinary skill in the art will appreciate, the acronym CNC refers to computer numerical control machines including, for example, computer controlled lathes.

Figure 9 shows a DIO module 3000, with similar components to Figure 8 bearing identical reference numbers. The DIO module includes a Digital I/O interface 3017 which transmits and receives digital data. In the embodiment shown, the DIO module

10 3000 transmits and receives 22 bits of data, with 8 bits dedicated to input, and the remaining 14 configurable as either input or output ports. The module can similarly be constructed to support greater or fewer bits, for example, 2, 8, 16, or 32 bits. In addition, the module 3000 may include one or more trigger lines for triggering reading or writing of data through the interface 3017. Exemplary downstream devices which

15 can be controlled via the module 3000 include input type devices such as digital multimeters and line condition monitoring devices such as proximity sensors, limit sensors, and output type devices such as relays. In general, the module 3000 can be used to control any device that accepts as input high/low logic signals, and can monitor any device which generates high/low logic signals. In cases in which the high/low logic

20 signals does not use the same high/low voltage level as the interface 3017, an intermediate device can be used to translate the high/low logic signals to and from the appropriate levels.

Figure 10 shows a DAC Module 4000 with similar components to Figure 8 bearing identical reference numbers. The DAC module 4000 is an output type device which

25 includes a Digital to Analog converter 4017 which converts digital (binary) data received from the I/O port 2023 into equivalent scaled voltages. As an example, a 12-bit DAC 4017 programmed to generate analog signal in the range of 0 to 10 volts would accept 4096 values ( $2^{12} = 4096$ ), from 0 to 4095, which it would convert into 4096 analog values from 0 to 10 volts in increments of  $10/4095$  volts. Exemplary

downstream devices which can be controlled via the module 3000 include, for example, servo motors, power supplies, current controllers, or any other device which requires and analog input control voltage.

Figure 11 shows a ADC Module 5000 with similar components to Figure 8 bearing identical reference numbers. The ADC module 5000 is an input type device which includes an analog-to-digital converter 5017 (ADC) which converts an analog signal into a binary digital value. As an example, a 12-bit ADC 5017 programmed to receive an analog signal in the range of 0 to 10 volts would convert a received analog signal into 4096 12-bit digital values in increments of  $10/4095$  volts, and a 12 bit ADC configured to convert an analog signal in the range of 0.1 to 200 volts would convert a received analog signal into 4096 12 bit digital values in increments of  $200/4095$  volts. A particularly preferred ADC 5017 is a 12 bit ADC which accepts signals in the range of 0.1 to 200 volts, has a sampling rate of about 200,000 Hz, gain of 2, 20, 200, and eight channels, each of which can be configured as single ended or differential. Exemplary downstream devices which can be connected to this module include any device which generates an analog output, including, for example, strain gauges, pressure sensors, pressure transducers, torque sensors, photodiodes, or simply a pair of wires attached across a voltage potential.

Figure 12 shows a UART Module 5000 with similar components to Figure 8 bearing identical reference numbers. The UART module 6000 is an input and output type device which includes a UART chip 6017, or universal asynchronous receiver-transmitter. UART 6017 is of conventional construction and includes a transmitter section and a receiver section. The transmitter converts bytes of parallel data from I/O port 2023 into a serial stream of data bits, and the receiver accepts a serial stream of data and converts it into bytes of parallel data which are transferred to the I/O port 2023. Examples of UARTs include RS 232 transceivers and RS 485 transceivers. Exemplary downstream devices which can be controlled via the module 5000 include, for example, digital multimeters, scales, amplifiers, modems, and CMC machines.

Figure 13 shows a GPIB Module 6000 with similar components to Figure 8 bearing identical reference numbers. GPIB Module 6000 is an input and output type device and includes a GPIB chip 7017 which converts data to and from the GPIB protocol discussed above. Exemplary downstream devices which can be controlled via the module 5000 include, for example, digital multimeters, and GPIB laboratory instruments in general.

Figure 14(a) illustrates a system including an intermediate controller 8000 disposed between a plurality of embedded modules (in the illustration, modules 3000, 4000, 5000) and the DAQ 1, and Figure 14(b) illustrates graphically the organization of programs and sockets on the controller 8000. Intermediate controller 8000 includes a NIC 2002 (right side) coupled to the DAQ 1 via a LAN or via the Internet, a NIC 2002 (left side) coupled to modules 3000, 4000, and 5000 via an ethernet hub 9000. The hub 9000 could be an external hub (as shown), or alternatively, could be incorporated into the controller 8000. The controller 8000 is configured as a server, and includes a CPU, ROM, and RAM memory for transferring information between the computer 1 and the modules 3000-5000. Preferably, the controller is implemented with a Linux operating system in order to reduce storage requirements, and communicates with DAQ1 and the modules 3000-7000 via an IP protocol such as TCP/IP or UDP/IP.

At power-up, the controller 8000 transmits a multicast signal to query any connected modules (3000-7000). Each connected module (in this case 3000, 4000, and 5000) responds with a MAC address, which is a hard-coded address ID which is specific to the particular manufacturer and model (e.g. ACME Co, DIO module). In any event, referring to Figure 14(b), an application program A1 running on the controller 8000 then assigns a mock IP address to each connected module. In this regard, the IP address is a "mock" address in that it is not a valid Internet IP address but, rather, is an address provided merely for purposes of allowing the controller 8000 to communicate with the connected modules via TCP/IP protocol. For example, the controller could designate each connected module with consecutive IP addresses beginning with 10.0.0.1, 10.0.0.2, etc. In contrast, the controller 8000 itself has a valid IP address which is

fixed by the Internet service provider (ISP). This IP address can be obtained by the controller 8000 by sending a request for the IP address to the DAQ 1 using, for example, the reverse address resolution protocol (RARP). Alternatively, the IP address could be provided to the module by direct data input, for example, via a serial port.

- 5 The application program A1 then initiates a respective program having a respective "socket" (or other software connection) for each connected module, so that communication with each module is run independently. Application programs A1a, A1b, and A1c having sockets s1, s2, and s3 are used to provide an independent software connection (e.g. thread) between the modules 2000, 3000, 4000 and the controller
- 10 8000. The application program A1 also initiates an application A2, which, in turn, initiates respective programs A2a, A2b, and A2c having corresponding sockets s1', s2', and s3' for providing an independent software connection (e.g. thread) for communicating data from the modules to the DAQ system 1. Each of these sockets s1', s2', and s3' have a corresponding channel in the DAQ system 1, which channels operate
- 15 in the same manner as described above. In addition, each socket s (e.g. s1, s2, s3) has a corresponding buffer a (e.g. a1, a2, a3), and each socket s' (e.g. s1', s2', s3') has a corresponding buffer b (e.g. b1, b2, b3). As one of ordinary skill in the art will appreciate, a socket is defined as the combination of an IP address and a port number, and is used in the TCP/IP protocol to direct data to the appropriate application in the
- 20 network. The IP address for the sockets s1-s3 are the mock IP addresses discussed above, whereas the IP address for the sockets s1'-s3' is the IP address of the controller 8000. The port numbers for sockets s1-s3 are generated by the program A2. The port numbers for the sockets s1'-s3' are preferably assigned by program A2 in accordance with conventional IANA assigned "Well Known Port" numbers (e.g., 20 for FTP, 23
- 25 for Telnet, 25 for smtp, and 80 for http). The port number and module type for each connected module (2000-7000) is transmitted to the DAQ 1 so that the DAQ 1 program can identify the connected modules.

When data is received from a module on a socket s1, for example, the data is stored in buffer a1, and an IPC is generated to alert application a2a that data is available to

transmit via socket s1'. Application a2a then fetches the data from buffer a1, and transmits it to the DAQ 1 via socket s1'. Similarly, when data is received for module 2000 from DAQ1 over socket s1', application a2a stores the data in buffer b1, and generates an IPC to application a1a indicating that data is available. Application a1a then fetches the data from buffer b1, and transmits it to the module 2000 via socket s1.

In accordance with another embodiment of the present invention, the modules 2000 to 7000 can be directly connected to the DAQ 1 without the use of a controller 8000. In accordance with this embodiment, the individual module (2000-7000) is configured to communicate over an IP protocol using a valid IP address, assigned, for example, by an ISP. Preferably, the module is programmed to communicate to the remote DAQ1 via NIC over TCP/IP protocol. In this regard, an application on the module (2000-7000) establishes a socket using the IP address and a designated port. As set forth above, the port is preferably assigned in accordance with conventional IANA assigned "Well Known Port" numbers. The IP address for the modules can be assigned in a variety of ways, but is preferably obtained by sending a request to the DAQ 1 for an IP address using, for example, the reverse address resolution protocol (RARP). Alternatively, the IP address could be provided to the module by direct data input, for example, via serial port 2012.

#### Remote Capability

The use of the shared memory 201 through 204 allow the data-input-type and data-output-type devices to be controlled, during run-time, from remote computers. In this regard, referring to Figure 2, the run-time control values stored in Shared Mem DLLs 204 and 203 and can be accessed and changed remotely over a computer network as indicated by components Network 101 and Network 104 respectively. Similarly, data received from an input-type device can be accessed remotely from Shared Mem DLL 201 as indicated by service routine To Network 110. In addition, data to be transmitted to an output-type device can be input remotely to Shared Mem DLL 201 as indicated by service routine From Network 116. The network-related service routines can be implemented using object-oriented remote procedure call. To facilitate reliable

machine-to-machine communications, much of the e-mail messaging infrastructure provided by the Internet can be adopted. This can be implemented, for example, with TCP/IP protocols. Communication over a Local Area Network (LAN) can similarly be implemented via Ethernet and the TCP/IP protocols in a conventional manner.

- 5 In accordance with a preferred embodiment of the present invention, a remote operations control system (ROCS) is provided to allow users to remotely administrate hardware devices 2, 4, 2000-7000 attached to the system. The remote administration of hardware can be done either directly from DAQ system 1 or through the intermediate controller 8000. The implementations and procedures for the ROCS can be subdivided
- 10 into the following categories:

DAQ System 1 Software Based

DAQ System 1 Java Based

Intermediate Controller 9000 Software Based

Intermediate Controller 9000 Java Based

- 15 As indicated above, there are two modes of remote operation: 1) a remote client computer which accesses the DAQ system 1 to control devices 2, 4, 2000-7000 connected to the system 1; 2) a remote computer which communicates directly with an intermediate controller 9000 to control the devices 2000-7000 which are connected to the intermediate controller 9000.

- 20 In order to understand the specifics of the ROCS it is helpful to review the underlying configuration of an exemplary system and its corresponding components. Such a system is illustrated in Figure 15. For the purposes of this example there are three fundamental classes of components that are required for the remote operations control system: devices, master control unit, and a client:

- 25 Devices 2, 4, 2000-7000. As discussed above, these are hardware components connected to the DAQ system computer 1 (See Figures 1 and 15). These

components feed information into the DAQ system 1 and receive commands from it. Examples of the devices include multimeters, wave function generators (e.g. devices 4 of Figure 1), and the devices 2, 2000-7000 discussed above with regard to Figures 1, 8-13.

5 Master Control Unit (MCU): This is the computer 1 containing the authoritative copy of the DAQ system software and to which the hardware devices are physically connected. This computer acts as a server to the connecting clients. From this computer it is possible to monitor and control the attached devices.

10 Clients 5: These are the remote instances of software used to connect to the Master Control Unit for the purposes of monitoring or controlling the devices. The client software cannot act independent of the Master Control Unit in interfacing with the devices and must connect to a Master Control Unit in order to function.

15 To set up a remote operations control network with this configuration (as opposed to the intermediate controller configuration discussed below), at least one of each of the components mentioned above must be present. A device connects physically to the MCU through a hardware interface on the MCU. In this regard, the MCU is a computer 1 having installed thereon an instance of the MCU DAQ software. The computer 1 can interact with the devices through normal communications as described above with  
20 regard to non-remote operations. However, in addition to this core functionality, several additional features are added to allow for remote operations control of the devices. To accept remote operations control requests, the MCU is placed in the ROCS mode. Once ROCS mode has been established, a listening network socket is created on the network interface connected to the network from which the ROCS  
25 requests will originate. This socket is referred to as a ROCS Authorization Request Port. The port listens for a connection to be made to it by one of the clients. It is important to note that while the MCU is in a state to listen for ROCS requests, the MCU itself can preferably still function as the control authority over the devices on its

network (e.g., from a user entering keystrokes directly into the computer 1). However, once controlling authority of one or more devices has been passed from the MCU to a client, the MCU no longer has active control over those devices, preventing it from issuing commands to those devices (although it is able to passively monitor activity on the system from any and all devices).

When a client computer 5 (e.g. a computer implementing client DAQ software) wishes to gain ROCS privileges to the devices, it first makes a request to the ROCS Authorization Port. The purpose of this port is two fold. First, the ROCS Authorization Port is designed so that only one instance of the GDAQ software (either client or MCU) can control any given device connected to the MCU at any given time. If for example client A has authoritative control over device B, the MCU and all other clients will be refused control privileges over device B until client A has relinquished control back to the MCU. Preferably, however, the MCU, by virtue of its master authority over the system as a whole, is allowed to remove any remotely connected users and take control of any and all devices on the network. The second function of the ROCS Authorization Port is to verify the identity of the user making the request. The schemes used for this authentication can vary from username and password combinations to public key cryptosystems and digital signatures or combinations thereof. The type of security measures implemented in the system and extent to which security precautions are taken are a function of both the network on which the MCU and clients reside and the level of security required by the user. Once the client has been verified and it is possible for the control request to be granted by the system the connection protocol begins.

When an MCU detects that an authorized requesting client can be issued control of the modules several steps are taken, again many of which depend on the security precautions required by the user. Regardless of the security implementation, once a client is given approved by the MCU for control, the MCU allocates a series of ports on its network connection. A corresponding port is provided for each device the client wishes to interact with. It should be noted that it is not necessary for the client to have

controlling authority over the modules in order to initiate this procedure. For example, a client could request, and the MCU grant, the ability to monitor data from the device without granting any controlling authority over the device. In contrast to controlling authority, multiple clients can be provided with the ability to monitor data from device  
5 at the same time. Monitoring authority can be granted using a similar procedure to controlling authority without the ability to send commands.

In a passive security environment, the sockets are allocated for the client's connections and the client waits for those connections to be established. For a system with more elaborate security precautions, packet filtering rules can be modified to only allow  
10 incoming packets to those ports from IP addresses that match the client which was authorized for use on the MCU. In any event, once the security requirements are met, session begins and will continue to function until either the connection is lost or the client or MCU ends the session.

Incoming data to the client is in the form of statistically averaged data packets at a rate  
15 matching that at which the client is connected. Control packets are of a limited data size and will be sent over network channels. Therefore, the delay time in transmission is the only major factor that should be taken into account when using the system. Preferably, the system allows a client to request control of a device on an individual basis. In other words, it is not necessary to allocate control of all the devices  
20 simultaneously. This allows other remote clients or the MCU to retain control over other devices on the network. The system can also include an option for "an all or nothing allocation scheme" in which all devices must be allocated to a single client. This option is useful for applications in which individual access to devices is dangerous (e.g., in an application in which all of the devices relate to closely inter-related  
25 processes).

As mentioned above, the system preferably allows multiple clients and the MCU to concurrently monitor data. This can be initiated by making a monitoring request to the MCU Monitoring Request Port. Preferably, this port can be set or removed at the

discretion of the MCU operators. In any event, through this port, security measures can either be implemented or ignored as described above with regard to the MCU Authorization Port scheme. This port performs the same allocations as described above before allowing any authorized user (or any user at all if there are no security checks) to

5 have a monitoring port opened to give them access to monitoring the data. Data that is segmented off to a shared memory region in the MCU DAQ software can therefore be sent not only to the terminal in which it is running (the MCU), but also can be sent from that shared memory region to one or all of the sockets connected to a client interface. The inclusion of the client sockets on the MCU simply changes the number of targets

10 for the shared memory region that holds the device data.

The client software used to interact with the MCU can be implemented in a number of ways. For example, client software could be provided in the form of a compiled program that must be installed on the client machine in order to run. This software can, in essence, be a limited version of the MCU DAQ software, which does not have the

15 server utilities. To run, this type of client DAQ software simply needs to be installed on the client computer and have a network connection to the MCU. Alternatively, the client software could be in the form of a Java applet. In this embodiment, the MCU would not only contain the listening sockets and authorization ports, it would also contain a port that upon connection uploads a Java applet on request to a client

20 computer. The applet sent to the client would be a client version of the DAQ software capable of being executed on any machine. This embodiment allows for any type of client environment that supports the Java VM. In addition to the GUI interface, a more basic text version could be provided to allow interfaces from console based machines or even PDA devices. The same type of authorization schemes can be set in place or none

25 at all, just as in the installable client DAQ software.

A second example of a remote communication system is one which uses the intermediate controllers 8000. Such a system is illustrated in Figure 16. In this example, a system includes:

Devices 2, 2000-7000: Hardware devices, such as RS-232C, DIO, DAC, and ADC modules which transmit and receive their data through a standard network interface.

5 Intermediate Controller 8000: A piece of software residing on an independent piece of hardware is coupled to the devices 2000-7000 via the standard network interface, and that performs control and data routing to and from the devices from the clients.

Clients 5: These are the remote instances of software used to control the modules via the intermediate controller 8000.

10 In this embodiment, the intermediate controller 8000 is substituted for the MCU as the intermediate component between the clients 5 and the one or more devices 2, 2000-7000 connected to the intermediate controller 8000. The intermediate controller is preferably implemented in a custom built piece of hardware (to minimize its footprint), but can also be installed on a conventional computer (such as a PC) that resides on the  
15 same network as the modules. The intermediate controller 8000 creates and tracks all the connections with the devices, and handles all communications between itself and the devices 2000-7000 on the network. The intermediate controller 8000 also maintains a connection to a network which the clients can access (e.g., the Internet, other WAN, or even a LAN). The clients make their connections and requests for control of the  
20 devices 2000-7000 through the intermediate controller 8000. In this manner, the intermediate controller 8000 acts as a control router for the devices 2000-7000. Similar to the MCU, the intermediate controller 8000 can have ROCS Monitoring Request and Authorization Ports, and can create listening sockets to handle incoming connections. Like the MCU, the intermediate controller 8000 can incorporate a series of security  
25 measures such as username / password pairs or public key cryptosystems to ensure only authorized users are allowed to connect to the system and that there is only one user in control of a device at a time. In addition, the intermediate controller 8000, like the MCU, can have the capability to lock out all users or distribute access rights to users for

individual modules on a case by case basis at the discretion of an administrator of the intermediate controller.

Unlike the MCU, however, the intermediate controller cannot, by itself, control the devices 2000-7000. With the intermediate controller embodiment, users can only  
5 control the devices 2000-7000 via a client. Although a super user may be allowed to log into the intermediate controller for administrative purposes, the intermediate controller 8000 otherwise performs its functions independent of users on the system. The intermediate controller 8000 handles all connections to the devices and, through the use of statistical functions, can distribute data to the connecting clients at rates they  
10 are able to withstand on their connections. Control functions from the clients are received by the intermediate controller and routed to the devices for processing. The intermediate controller 8000 preferably has the ability to limit the number, type, and access of all users connected to it. In addition, packet filtering rules can be set up for security reasons and dynamically changed based on authorized incoming user  
15 connections.

As with the MCU embodiment, there are a number of ways in which a client can connect to the intermediate controller 8000. For example, an application program written specifically to interface with a module of a specific type can be used. This software will make a request to the authorization port if it wishes control of a module or  
20 the monitoring request port if it merely wishes to monitor. In either case, the client will then go through whatever security measures are in place (e.g. basic username / password combinations, digital keys, or no security at all). Once the control or monitoring authentication has been completed, a series of sockets will be allocated for the client to connect to (see Figures 14(a-b)). These sockets will give the client either  
25 control or monitoring channels for the modules it has requested, assuming of course it has access to them. As with the MCU embodiment, any number of clients may have access to monitor data from any number of devices 2000-7000, with the proper access rights, but only one client may have control over any given device at a given time. "All or nothing allocation schemes" can similarly be supported by the intermediate

controller. Control and monitoring can continue from this point until either the connection is lost or the client ends his session. Should either one of these events occur cleanup measures and reallocation of control rights will be returned to the system.

- Similar to the MCU embodiment, the client software for the intermediate controller
- 5   embodiment can come in a number of forms. For example, the client software can be provided in a compiled program that is installed on the user's computer (the client). The client software can then be executed to connect to the intermediate controller 8000 via a network connection (such as the Internet, other WAN, or even a LAN). Alternatively, the client software can be provided in the form of a Java applet or a Java application.
- 10   In accordance with this embodiment, the client may, through the use of a browser, connect to a port on the intermediate controller 8000, download the JAVA applet therefrom, and then run the JAVA applet through the Java VM on the client computer. This embodiment allows for portability over platforms not specifically envisioned by system developers, and allows the user to be able to download the client software and
- 15   execute it from any computer to which they have access. The intermediate controller can also include an option (selectable by the intermediate controller administrator), to enable or disable the ability to download the Java applet from the intermediate controller.

What is claimed is:

1. A computerized method for controlling at least one instrument coupled to a computer, comprising
  - a) selecting an I/O interface for communicating with an instrument;
  - b) inputting into a document, as text, a set of vendor-supplied instructions for said instrument;
  - c) inputting into the document, as text, a set of values for run time variables for said instrument;
  - d) translating the text in the document into digital signals for communicating with the instrument via the I/O interface; and
  - e) controlling the instrument via the translated digital signals, wherein said step of controlling includes one or more of the steps of transmitting commands to said instrument and receiving data from said instrument and transmitting commands and data to said instrument;wherein at least steps c, d, and e can be performed sequentially or concurrently.
2. The method of claim 1, wherein the step of inputting a set of run time variables further includes the steps of
  - creating, with a graphical user interface, a run-time window, the run-time window including input fields for entering each of the values for the run-time variables;
  - writing said set of values for the run-time variables into said document.
3. The method of claim 1, wherein the instrument comprises at least one embedded device and at least one downstream device, the at least one downstream device being coupled to the I/O interface via the embedded device wherein the instrument comprises at least one embedded device.
4. The method of claim 1, wherein the instrument comprises at least one DMM.
5. The method of claim 3, wherein said embedded device includes a parallel port

interface for coupling to the downstream device.

6. The method of claim 3, wherein said embedded device includes a DIO interface for coupling to the downstream device.
7. The method of claim 3, wherein said embedded device includes a digital to analog converter for coupling to an input of the downstream device.
8. The method of claim 3, wherein said embedded device includes an analog to digital converter for coupling to an output of the downstream device.
9. The method of claim 3, wherein said embedded device includes a serial port interface for coupling to the downstream device.
10. The method of claim 3, wherein said embedded device includes a GPIB interface for coupling to the downstream device.
11. A system for controlling at least one instrument comprising:
  - a) creating a shared memory;
  - b) selecting an I/O interface for communicating with an instrument;
  - c) inputting into a document, as text, a set of vendor-supplied instructions for said instrument;
  - d) inputting into the document, as text, a set of values for run time variables for said instrument;
  - e) storing the text of the document in the shared memory;
  - f) accessing the shared memory and translating, with an interpreter, the text of the document into digital signals for communicating with the instrument via the I/O interface; and
  - g) controlling the instrument via the translated digital signals, wherein said step of controlling includes transmitting commands to said instrument and receiving data from said instrument;

- h) storing the data received from the instrument in the second shared memory;
  - i) accessing the second shared memory, processing the data contained therein with a service routine.
- wherein at least steps c through i can be performed sequentially or concurrently.
- 12. The system of claim 11, wherein the instrument comprises at least one embedded device and at least one downstream device, the at least one downstream device being coupled to the I/O interface via the embedded device
  - 13. The system of claim 11, wherein the instrument comprises at least one DMM.
  - 14. The system of claim 12, wherein said embedded device includes a parallel port interface for coupling to the downstream device.
  - 15. The system of claim 12, wherein said embedded device includes a DIO interface for coupling to the downstream device.
  - 16. The system of claim 12, wherein said embedded device includes a digital to analog converter for coupling to an input of the downstream device.
  - 17. The system of claim 12, wherein said embedded device includes an analog to digital converter for coupling to an output of the downstream device.
  - 18. The system of claim 12, wherein said embedded device includes a serial port interface for coupling to the downstream device.
  - 19. The system of claim 12, wherein said embedded device includes a GPIB interface for coupling to the downstream device.
  - 20. The method of claim 11, wherein the step of inputting a set of run time variables further includes the steps of

creating, with a graphical user interface, a run-time window, the run-time window including input fields for entering each of the values for the run-time variables; writing said set of values for the run-time variables into said document.

21. The method of claim 11, wherein the service routine is a plotting routine.
22. The method of claim 11, wherein the service routine transmits the data to a remote computer over the Internet.
23. The method of claim 11, wherein the service routine is a printing routine.
24. The method of claim 11, wherein the service routine is a file saving routine.
25. The method of claim 21, wherein the plotting routine plots a plurality of signals on a single x-axis and a plurality of y-axes.
26. A system for controlling at least one instrument comprising:
  - a) creating a shared memory;
  - b) selecting an I/O interface for communicating with an instrument;
  - c) inputting into a document, as text, a set of vendor-supplied instructions for said instrument;
  - d) inputting into the document, as text, a set of values for run time variables for said instrument;
  - e) storing the text of the document in the shared memory;
  - f) accessing the shared memory and translating, with an interpreter, the text of the document into digital signals for communicating with the instrument via the I/O interface; and
  - g) controlling the instrument via the translated digital signals, wherein said step of controlling includes transmitting commands to said instrument, and accessing a second shared memory to obtain data to transmit to said instrument;
  - h) writing data to be transmitted to the instrument to the second shared memory

with a service routine;

wherein at least steps c through h can be performed sequentially or concurrently.

27. The system of claim 26, wherein the step of inputting a set of run time variables further includes the steps of  
creating, with a graphical user interface, a run-time window, the run-time window including input fields for entering each of the values for the run-time variables;  
writing said set of values for the run-time variables into said document.
28. The system of claim 26, wherein the service routine is a keyboard routine.
29. The system of claim 26, wherein the service routine receives data from a remote computer over the Internet, and writes the data to the second shared memory.
30. The system of claim 26, wherein the instrument comprises at least one embedded device and at least one downstream device, the at least one downstream device being coupled to the I/O interface via the embedded device
31. The system of claim 26, wherein the instrument comprises at least one DMM.
32. The system of claim 30, wherein said embedded device includes a parallel port interface for coupling to the downstream device.
33. The system of claim 30, wherein said embedded device includes a DIO interface for coupling to the downstream device.
34. The system of claim 30, wherein said embedded device includes a digital to analog converter for coupling to an input of the downstream device.
35. The system of claim 30, wherein said embedded device includes an analog to digital converter for coupling to an output of the downstream device.

36. The system of claim 30, wherein said embedded device includes a serial port interface for coupling to the downstream device.
37. The system of claim 30, wherein said embedded device includes a GPIB interface for coupling to the downstream device.
38. The system of claim 30, wherein the embedded device is coupled to the I/O interface via a local area network.
39. The system of claim 30, wherein the embedded device is coupled to the I/O interface via a wide area network.
40. The system of claim 30, wherein the wide area network is the Internet.
41. The system of claim 39, wherein the embedded device includes a network interface card, a memory, and a central processing unit, wherein the I/O interface includes a network interface card, and wherein the embedded device communicates with the I/O interface via an Internet protocol.
42. The system of claim 30, wherein the embedded device comprises a plurality of embedded devices, each coupled to at least one downstream device, and wherein the system further comprises an intermediate controller coupled between the plurality of embedded devices and the I/O interface, the intermediate controller connected to the plurality of embedded devices via a local area network, the intermediate controller connected to the I/O interface via a wide area network.
43. The system of claim 30 wherein  
the I/O interface includes a network interface card;  
the intermediate controller includes a network interface card, a memory, and a central processing unit, the intermediate controller communicating with the I/O

interface via an Internet protocol

each embedded device includes a network interface card, a memory, and a central processing unit, each embedded device communicating with the intermediate controller via an Internet protocol.

44. The method of claim 12, wherein the embedded device is coupled to the I/O interface via a local area network.
45. The method of claim 12, wherein the embedded device is coupled to the I/O interface via a wide area network.
46. The method of claim 12, wherein the wide area network is the Internet.
47. The method of claim 43, wherein the embedded device includes a network interface card, a memory, and a central processing unit, wherein the I/O interface includes a network interface card, and wherein the embedded device communicates with the I/O interface via an Internet protocol.
48. The method of claim 3, wherein the embedded device is coupled to the I/O interface via a local area network.
49. The method of claim 3, wherein the embedded device is coupled to the I/O interface via a wide area network.
50. The method of claim 3, wherein the wide area network is the Internet.
51. The method of claim 47, wherein the embedded device includes a network interface card, a memory, and a central processing unit, wherein the I/O interface includes a network interface card, and wherein the embedded device communicates with the I/O interface via an Internet protocol.

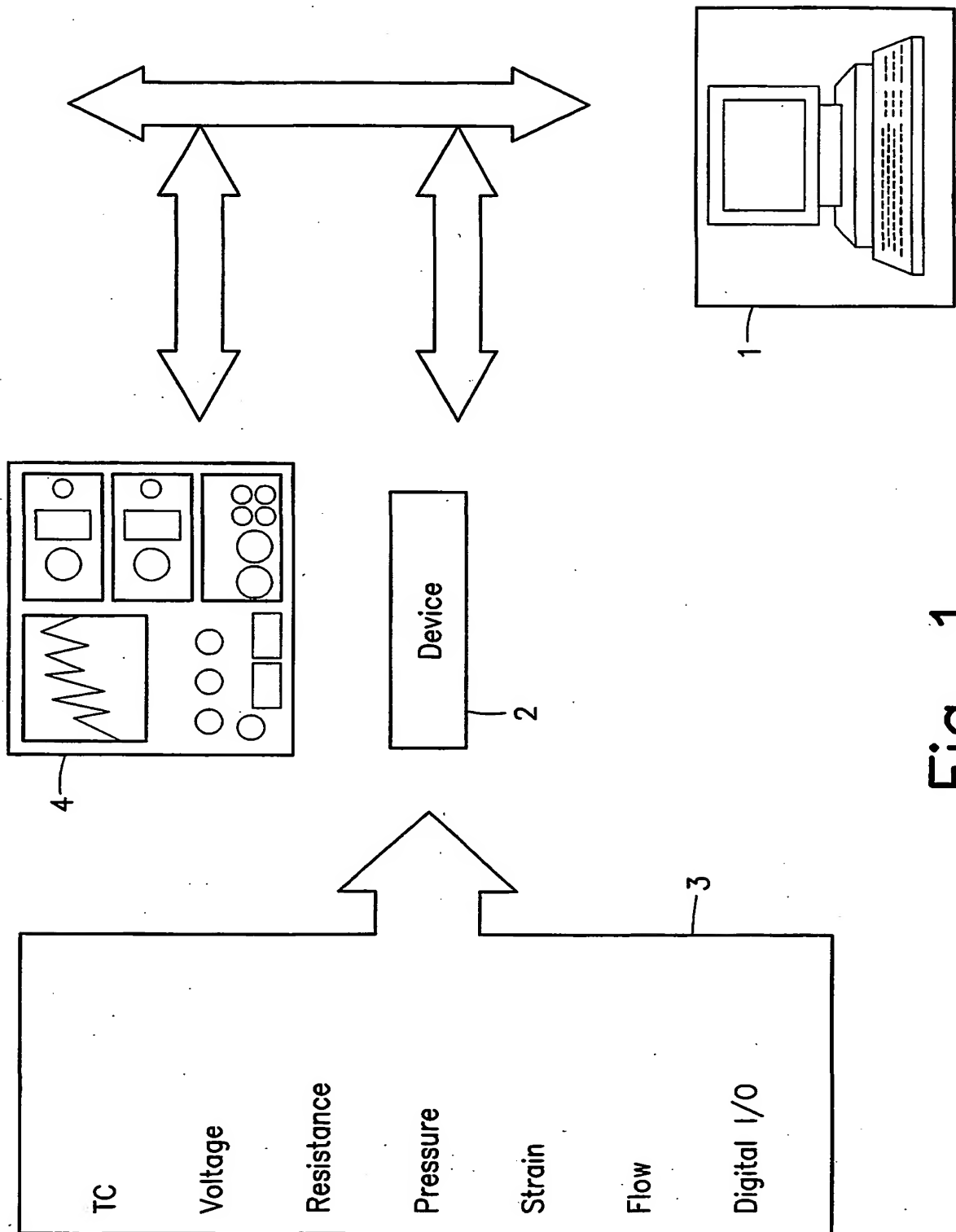
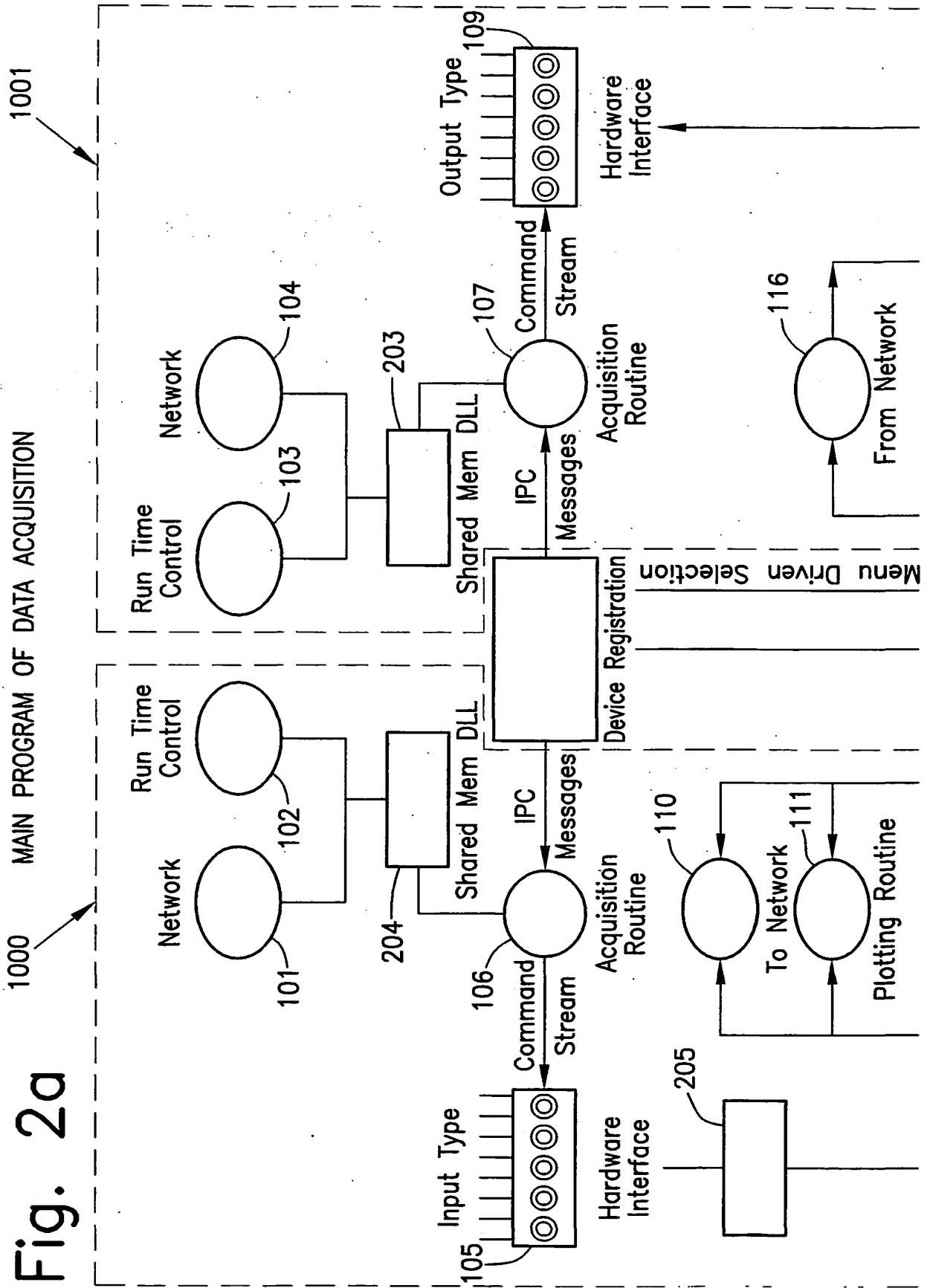


Fig. 1

Fig. 2a



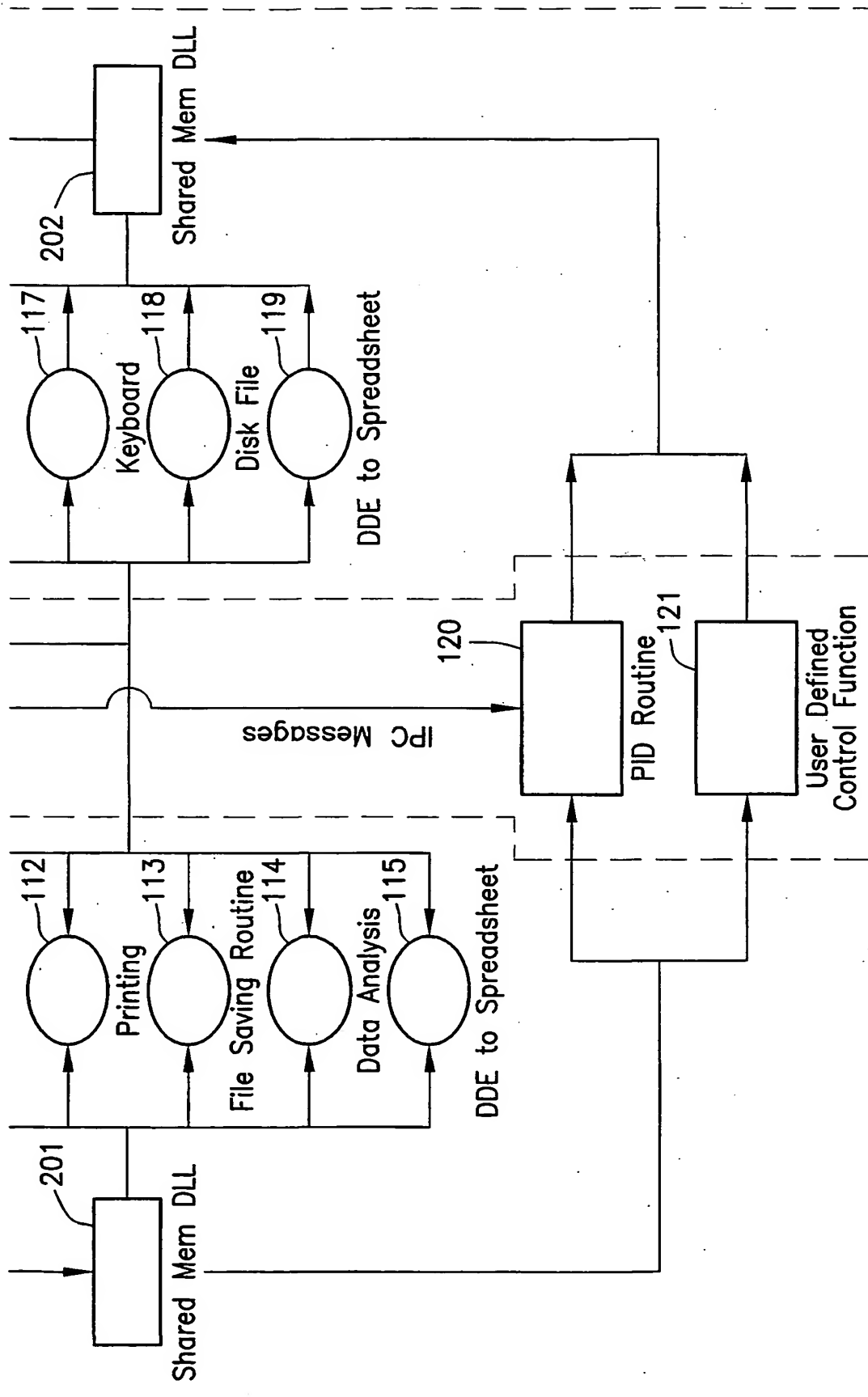


Fig. 2b

323 324 326 327 361 330 332 334 336

337 325 360 328 329 331 333 335

315 316 317 318 319 320 321 322

300 Main Code for

Input Type

Triggered by 1

305 Register Dev

Dev1 ☒ Dev2 ☒ Dev3 ☒ Dev4 ☐ Dev5 ☐ Dev6 ☐

Interaction-With-Dev

Receiving Data ☐ Sending Data ☐

306

	GPB Command	Target Val	Label	Contd
1	i Info Average No: \$x0\$	\$x0\$ 3	Ave No	3
2	s ibwr:rst:form:elem read:chan			
3	s ibwr:init:cont off:abor			
4	s ibwr:arm:lay1:sour imm:arm:lay2:sour			
5	w ibwr:routscan:intfunc \$x1\$,"RES"	\$x1\$ (@1,2,3)	Channel	1.45
6	w ibwr:sens:res:nplc \$x2\$	\$x2\$ 1	NPLC	0.01
7	w ibwr:sens:res:average \$x3\$	\$x3\$ 1	Average	0 2 4 6 8 10
8	w ibwr:routscan:intfunc \$x4\$,"volt:dc"	\$x4\$ (@4,5)	Channel	2.3
9	w ibwr:sens:volt:dc:nplc \$x5\$	\$x5\$ 1	NPLC	0.90
10	w ibwr:sens:volt:dc:average \$x6\$	\$x6\$ 1		
11	w ibwr:routscan:int \$x7\$	\$x7\$ (@1,2,3,4,5)		
12	w ibwr:trig:sour imm:count \$x8\$	\$x8\$ 5		
13	w ibwr:trac:feed sens			
14	w ibwr:trac:points:auto 1			
15	w ibwr:abort:trac:feed:cont next			
16	w ibwr:init:trac:data?			
17	r librd ReadStr:ReadNo			

Dev GPIB Add	Device ID	Signal #
4	1	5
First TP	2	Second TP
		Third TP
	15	16

Grid: Open Grid Save Grid SaveAs

GPB Command Test

Manual C Auto 300

Send Start Stop

Actual Test

Elapsed Time

Start Stop

GPB

Dev2 / Dev3 / Dac1 / Dac2

sens.volt:dc:nplc 0.90

310 311 312 313 314 350

Fig. 3

5/17

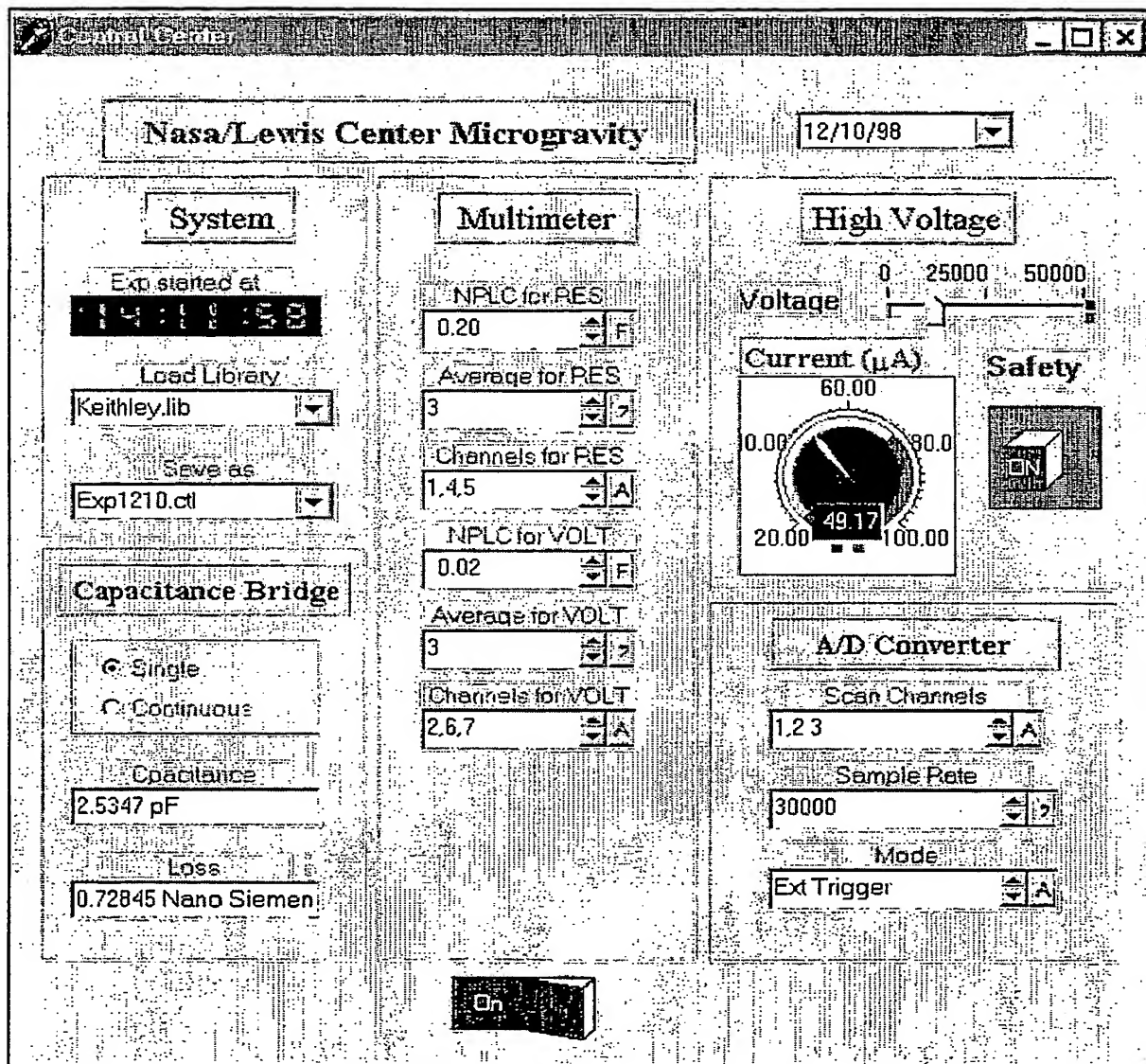


Fig. 4

6/17

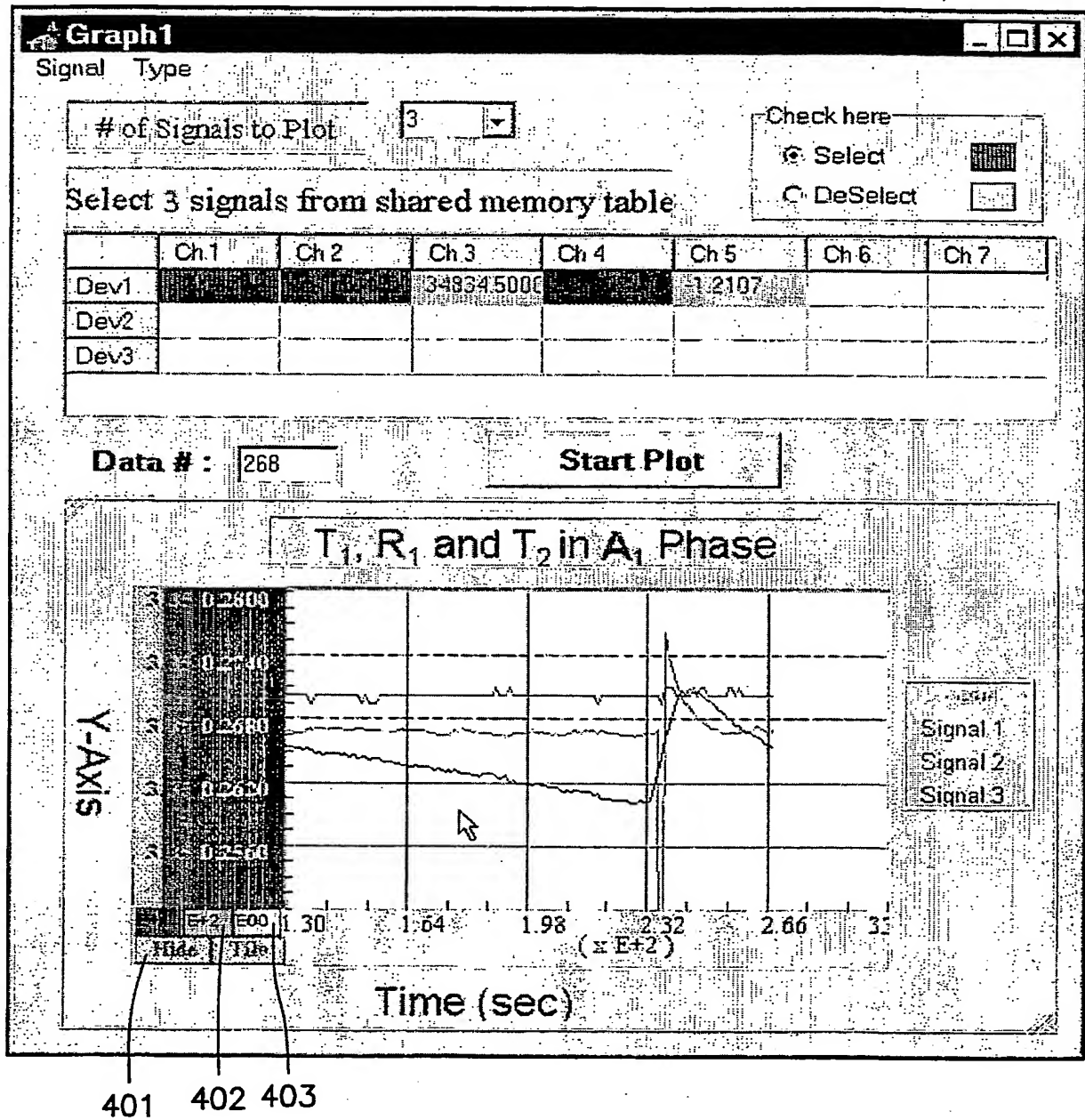


Fig. 5

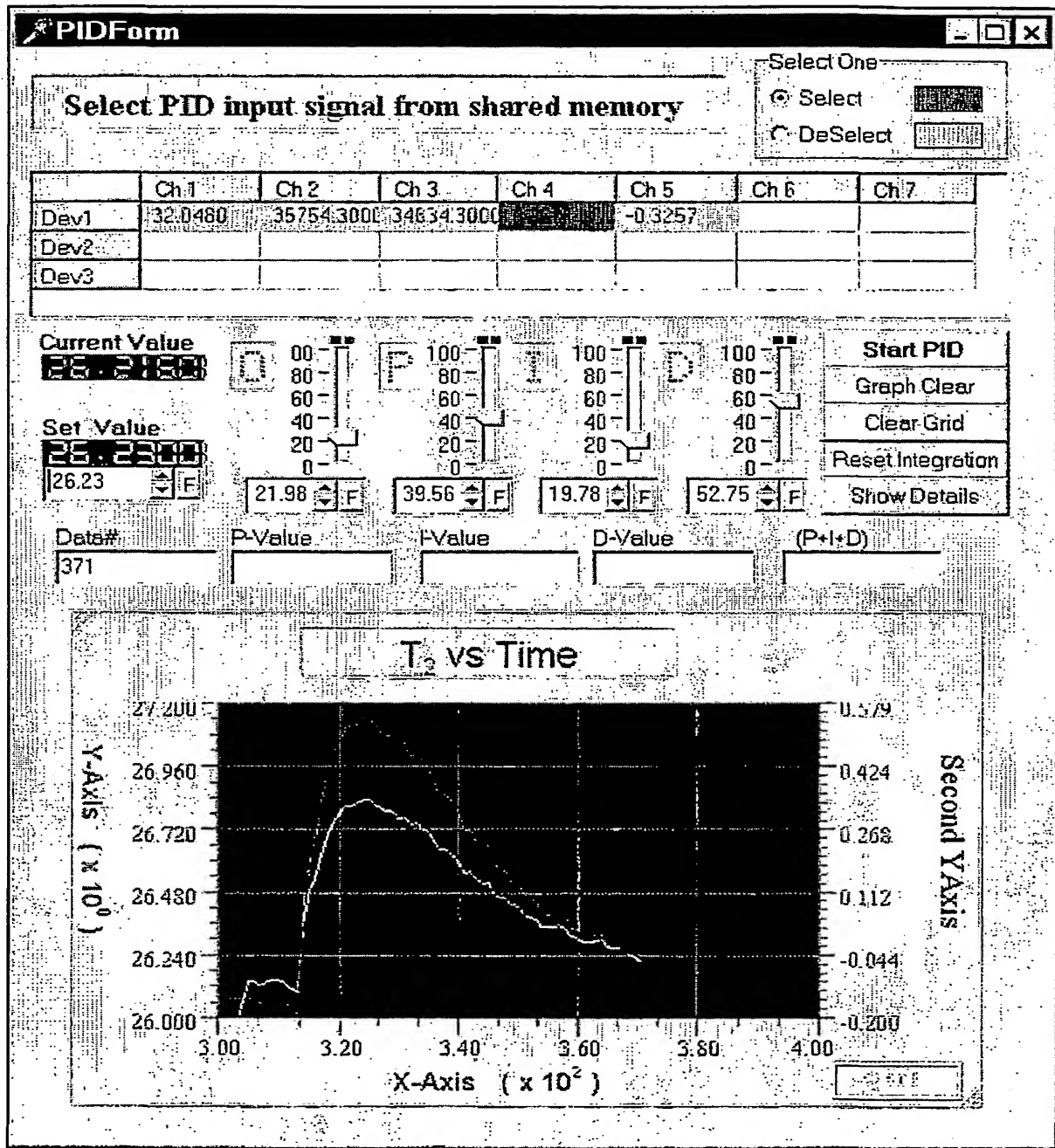


Fig. 6

8/17

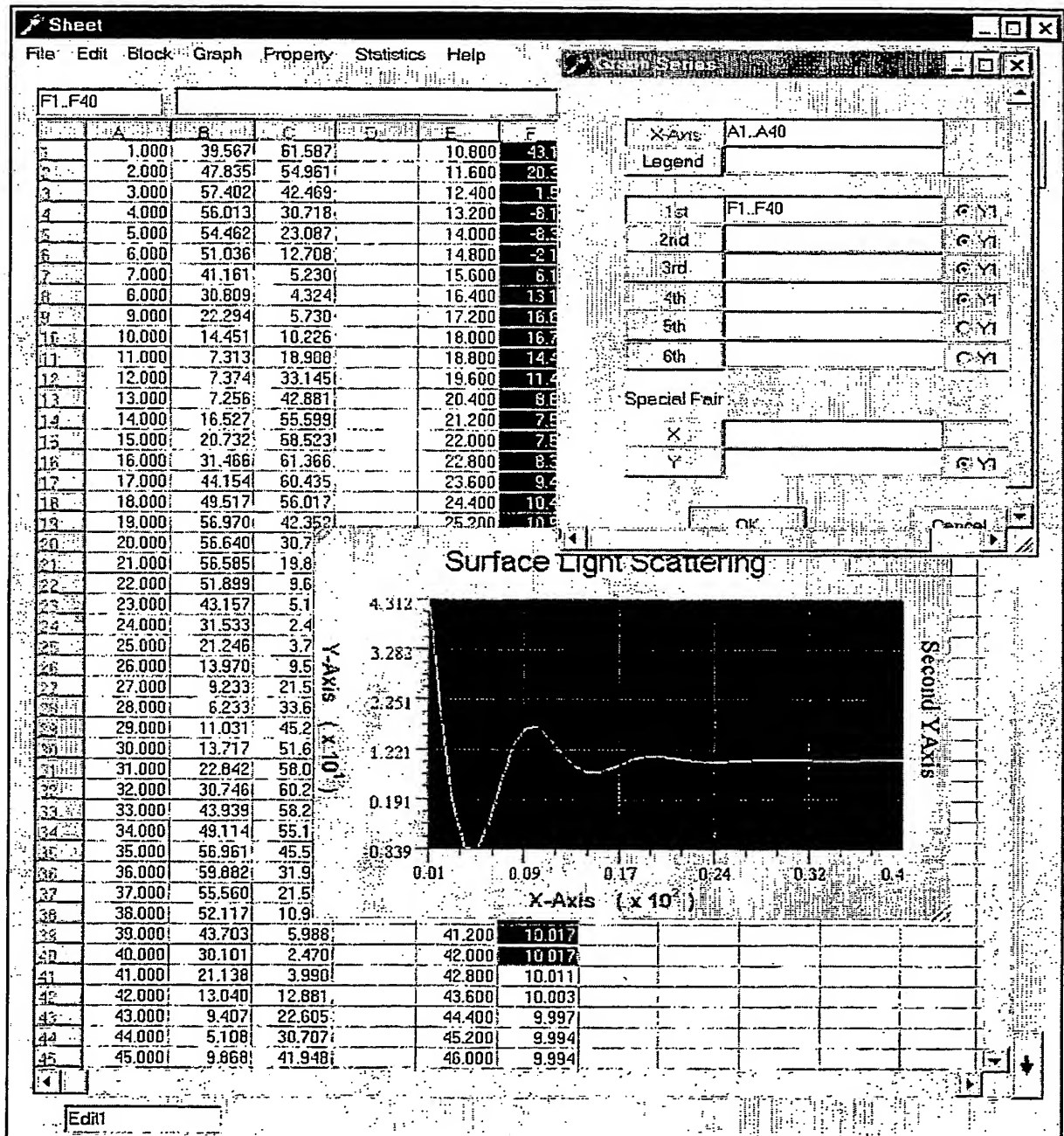


Fig. 7

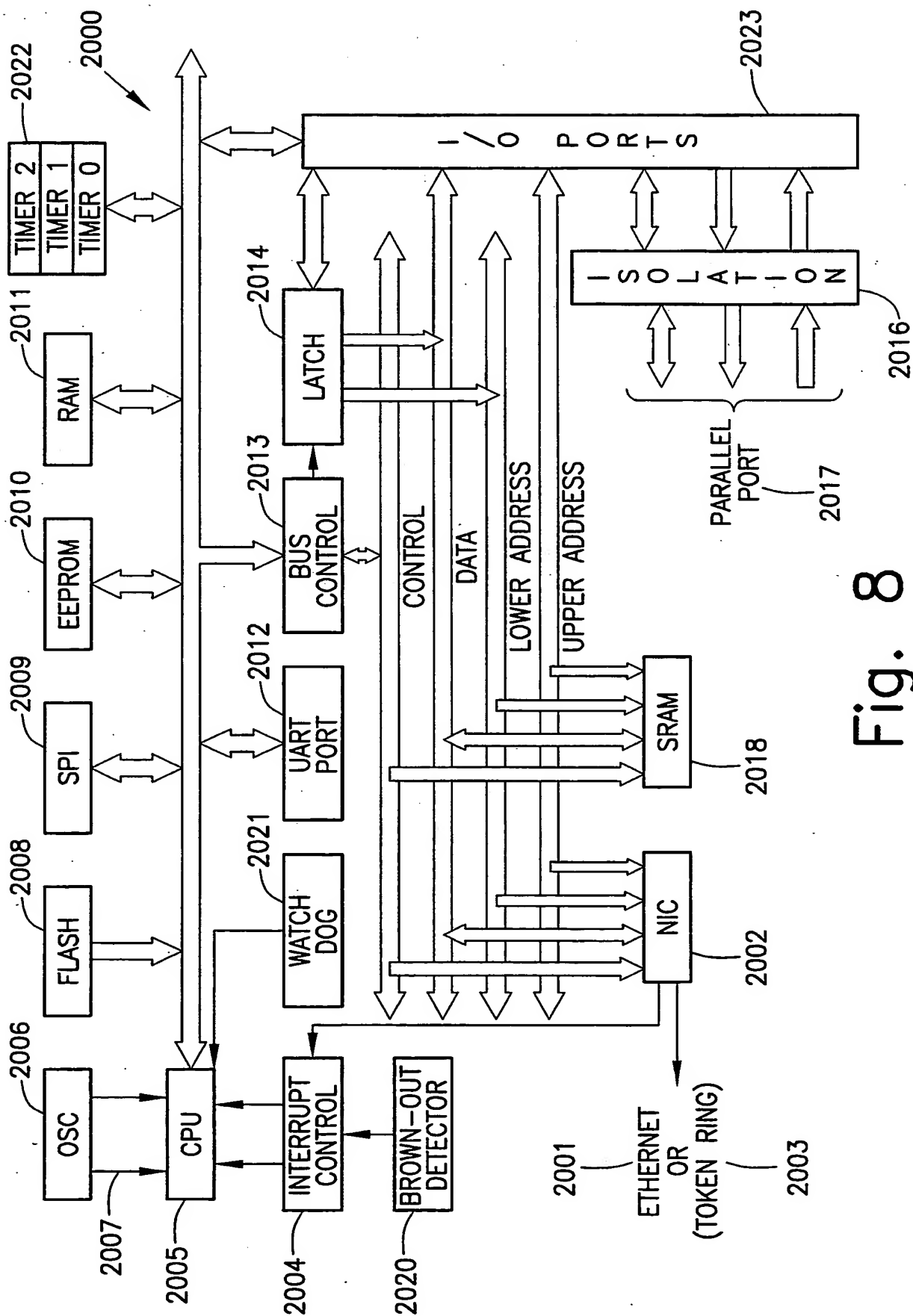


Fig. 8

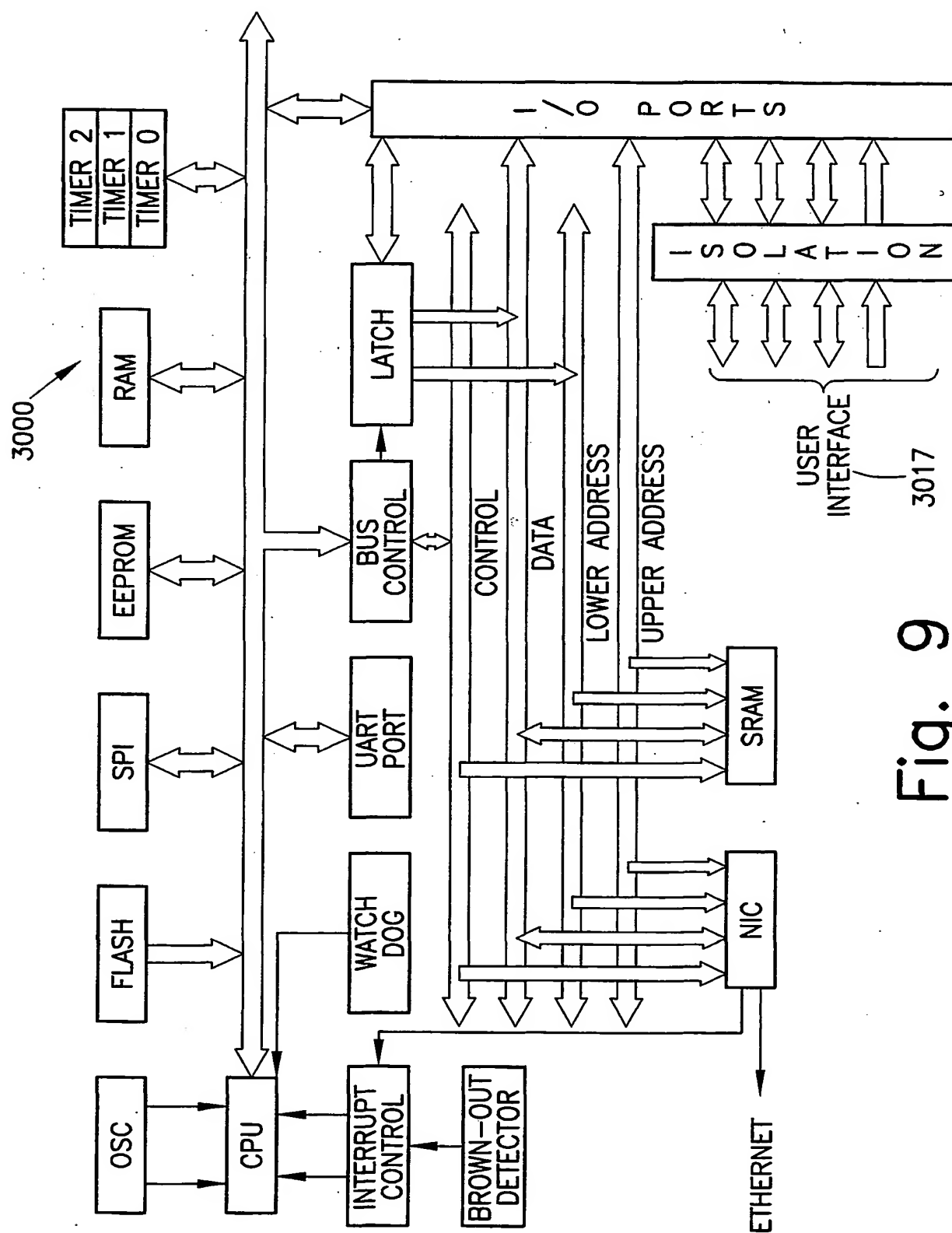


Fig. 9

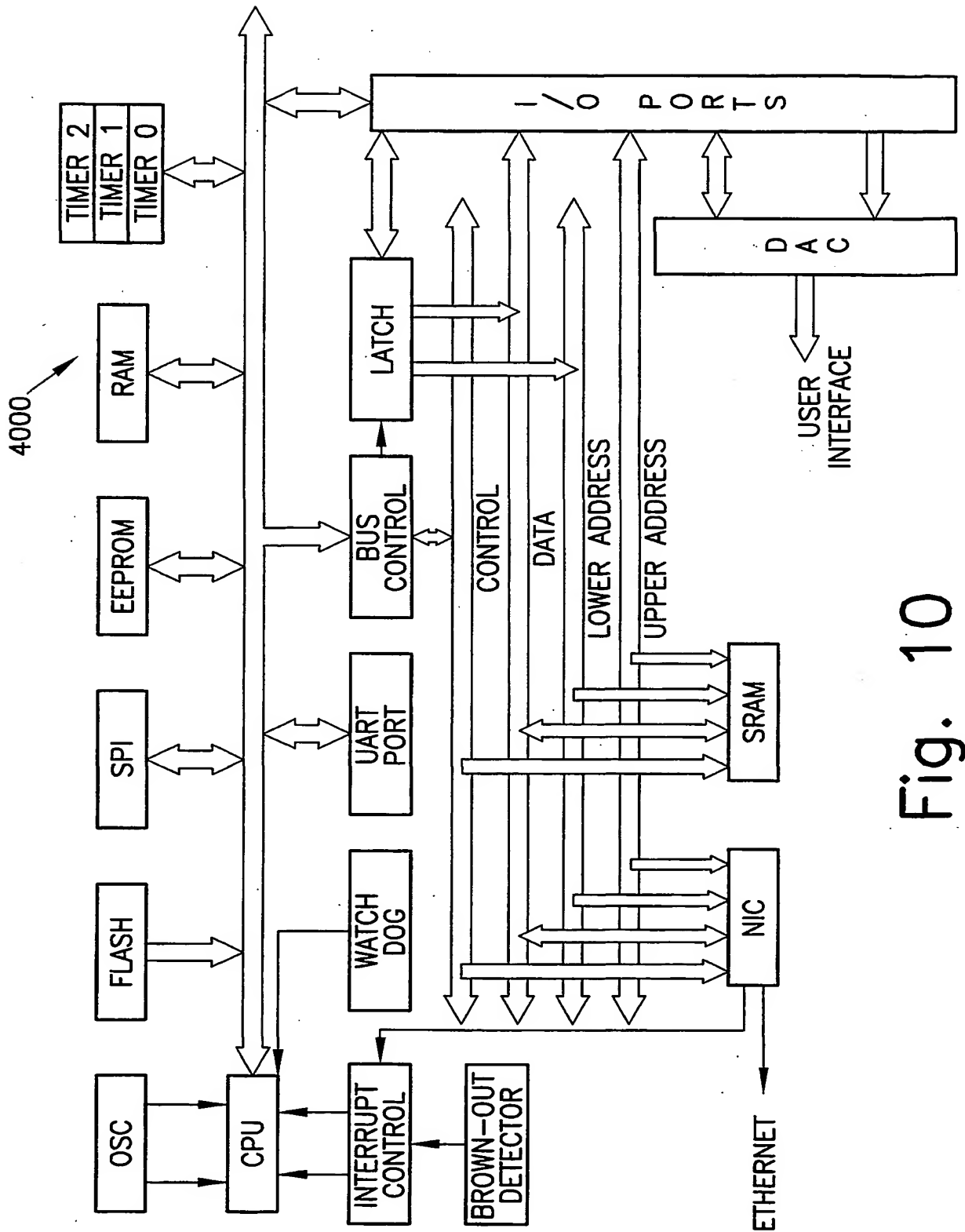


Fig. 10

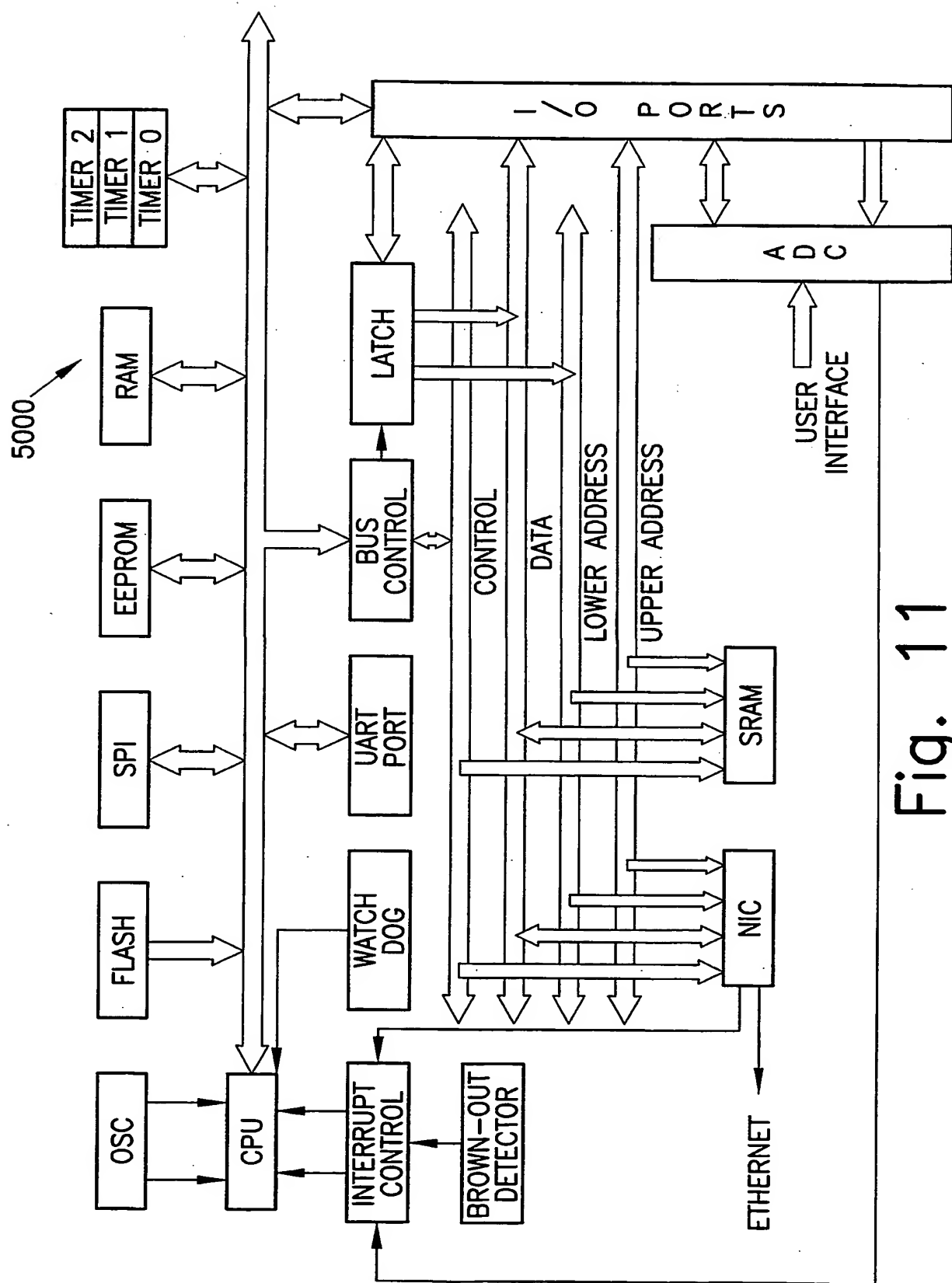


Fig. 11

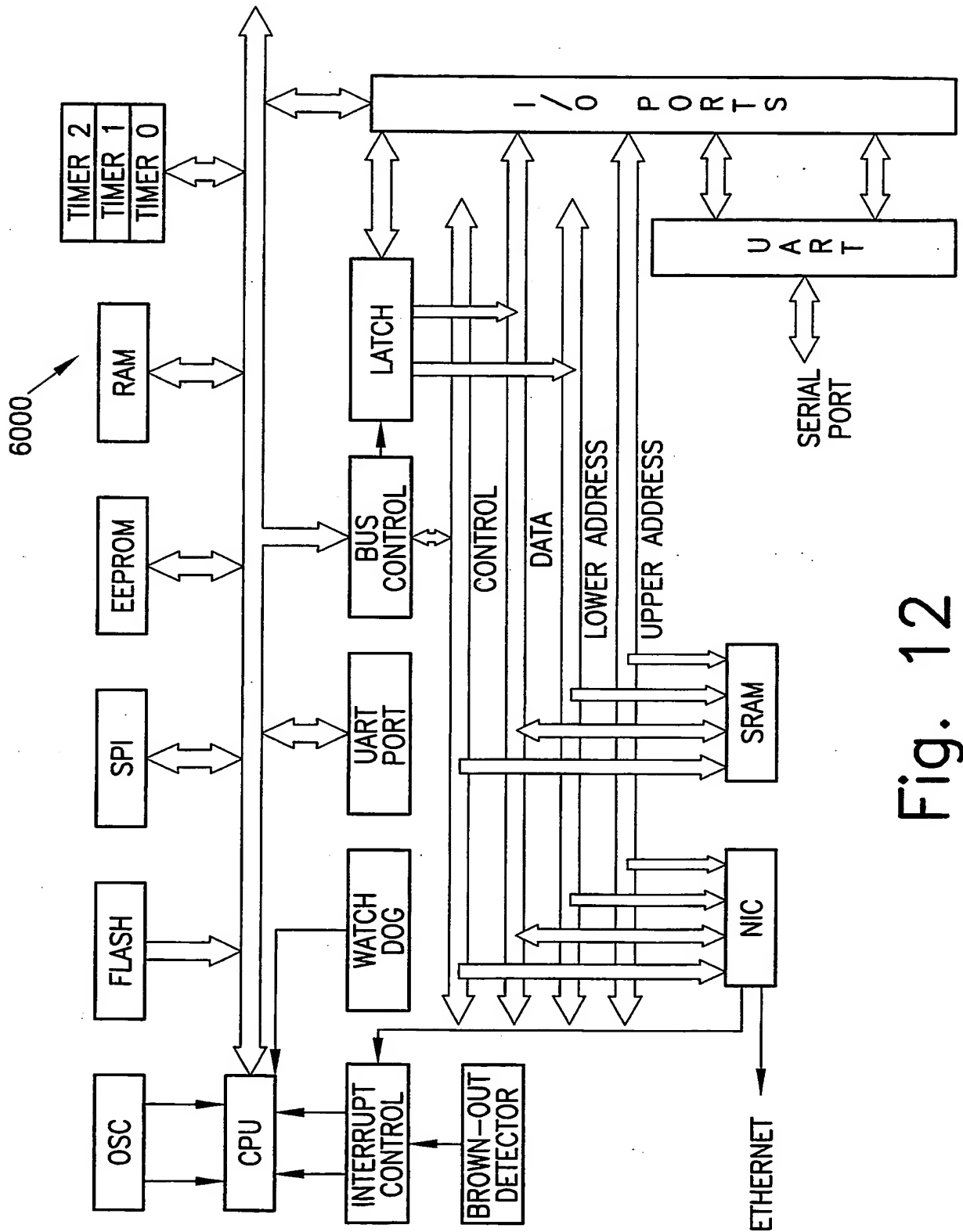


Fig. 12

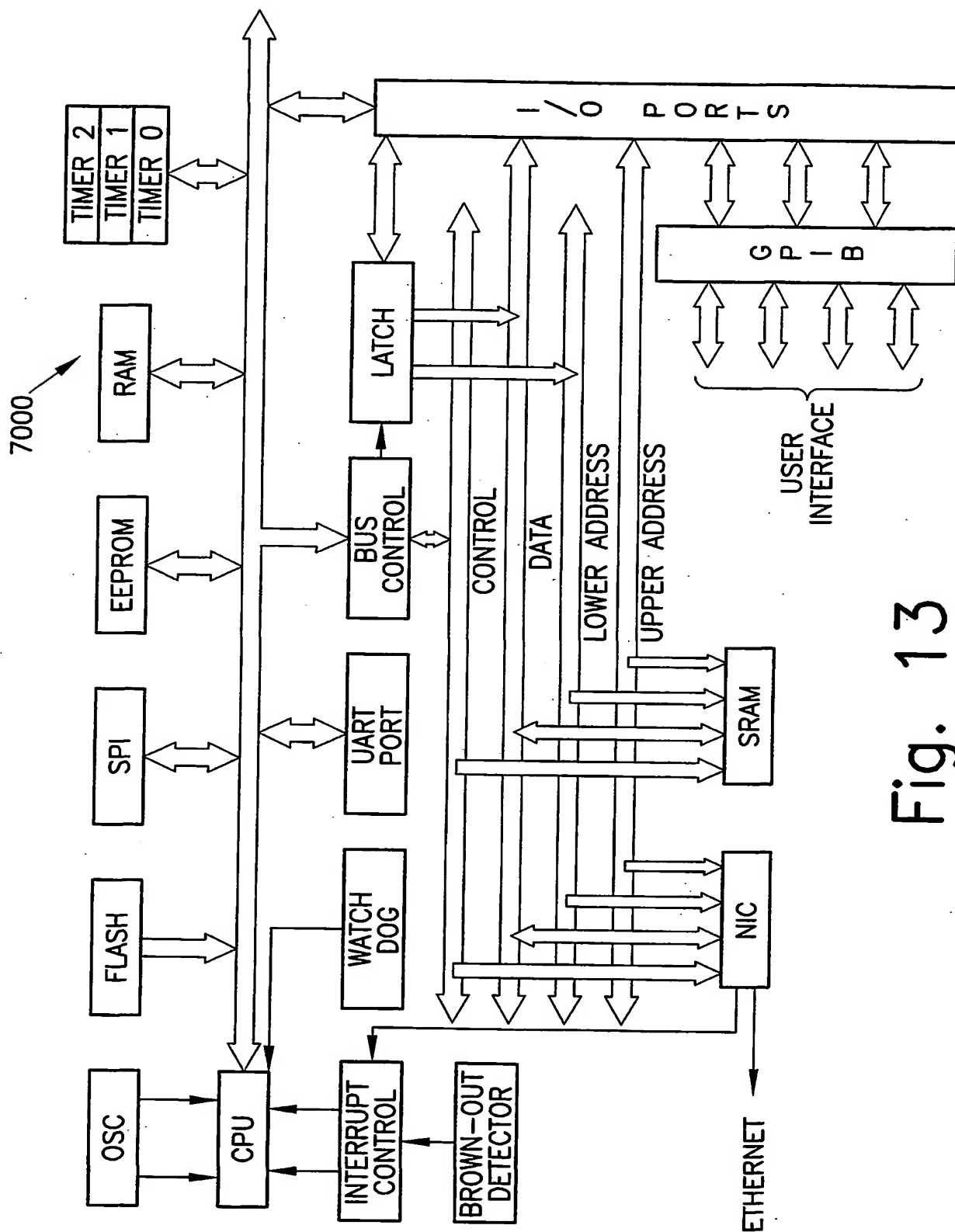


Fig. 13

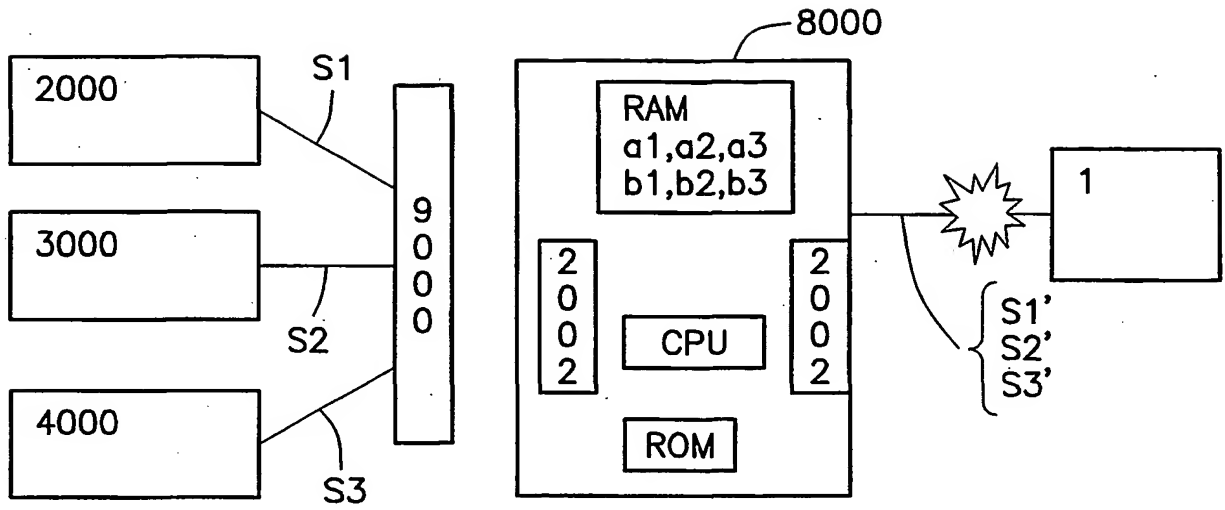


Fig. 14a

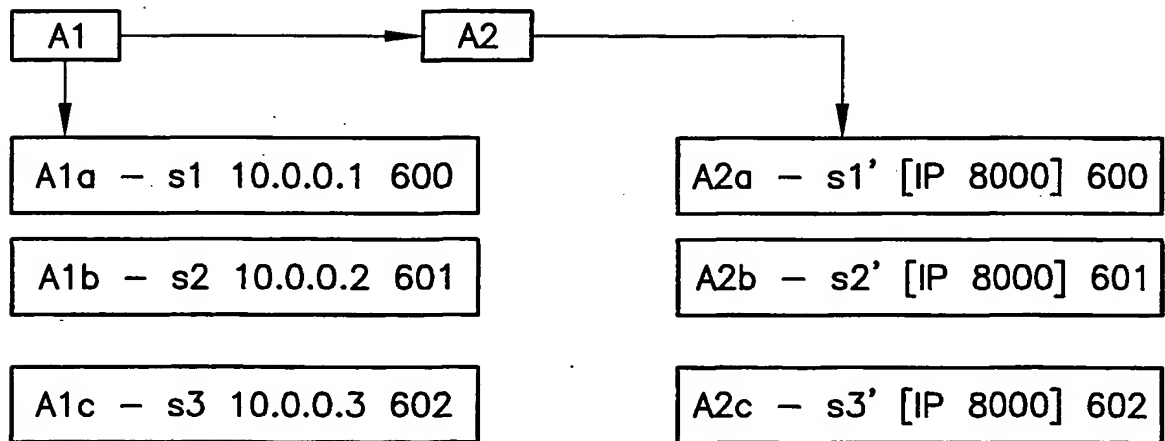


Fig. 14b

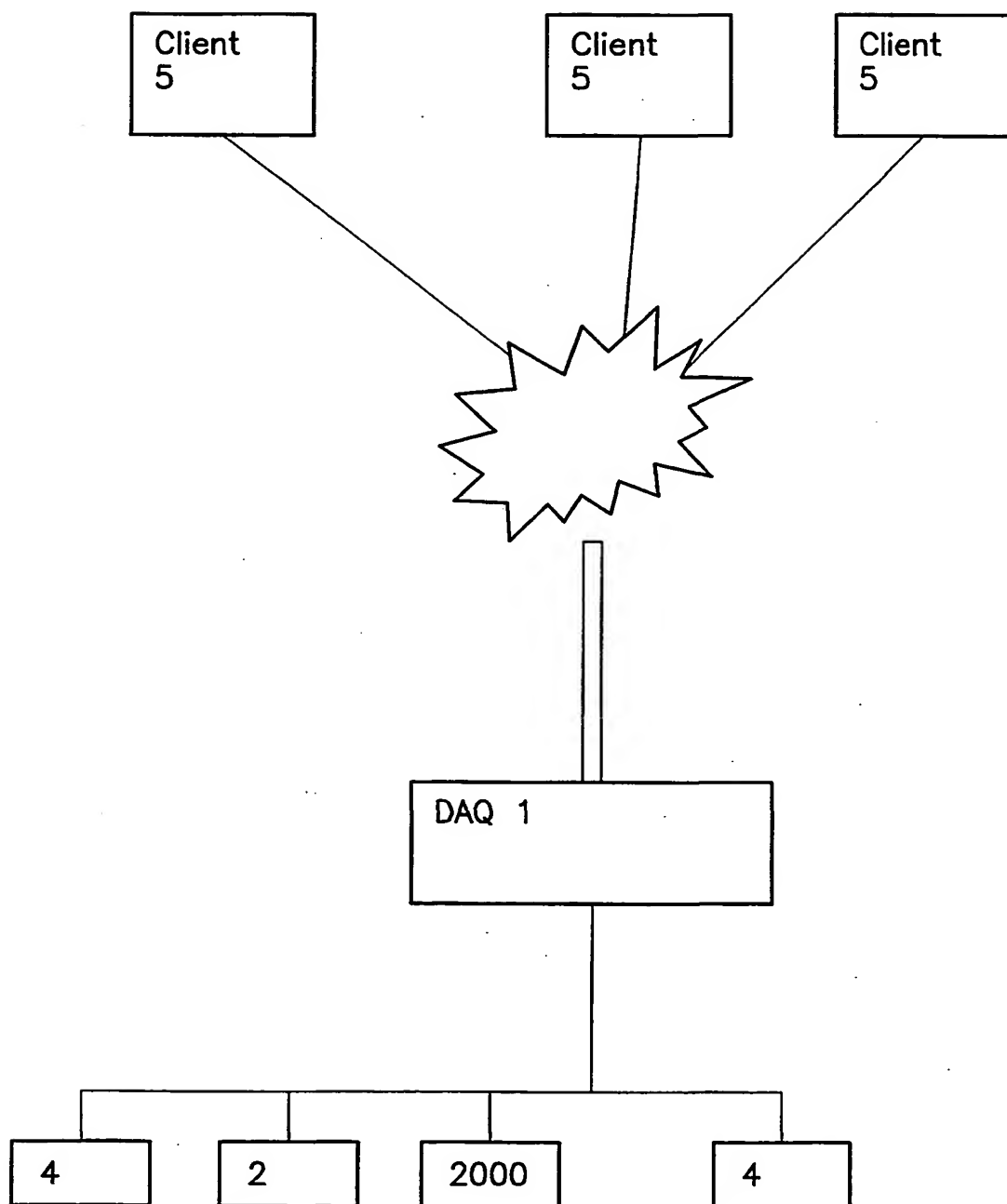


Fig. 15

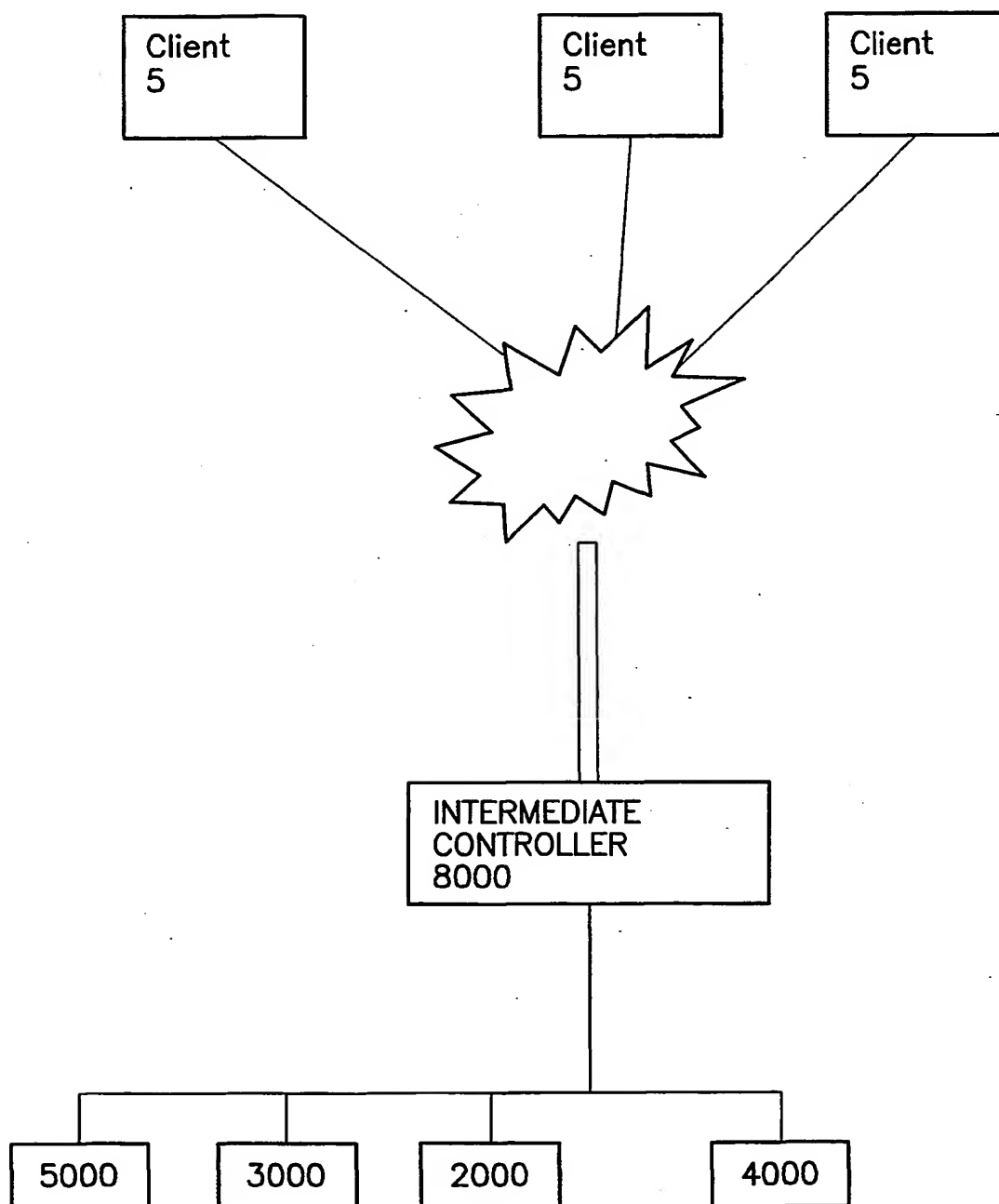


Fig. 16

## INTERNATIONAL SEARCH REPORT

International application No.  
PCT/US01/04277

## A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : G01C 17/38

US CL : 702/123

According to International Patent Classification (IPC) or to both national classification and IPC

## B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : Please See Extra Sheet.

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

NONE

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EAST

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X -- Y	US 5,926,775 A (BRUMLEY et al.) 20 July 1999 (20.07.1999), col. 5, lines 51-60.	1, 3-19, 22-24, 26, 28-51 ----- 2, 20, 21, 25, 27
Y	US 5,971,581 A (GRETТА et al.) 26 October 1999 (26.10.1999), Fig. 42.	2, 20, 27
Y	US 6,049,298 A (KNUDSEN) 11 April 2000 (11.04.2000), Figs. 18-24.	21, 25
A	US 5,764,546 A (BRYANT et al.) 09 June 1998 (09.06.1998), see entire document.	1-3, 5, 6, 9-14, 18-20, 23, 24, 26-28, 30, 32, 36, 37

☒ Further documents are listed in the continuation of Box C.
 ☐ See patent family annex.

* Special categories of cited documents:	*T* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
*A* document defining the general state of the art which is not considered to be of particular relevance	*X* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
*E* earlier document published on or after the international filing date	*Y* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
*L* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	*G* document member of the same patent family
*O* document referring to an oral disclosure, use, exhibition or other means	
*P* document published prior to the international filing date but later than the priority date claimed	

Date of the actual completion of the international search

14 APRIL 2001

Date of mailing of the international search report

03 MAY 2001

Name and mailing address of the ISA/US  
Commissioner of Patents and Trademarks  
Box PCT  
Washington, D.C. 20231

Facsimile No. (703) 305-3230

Authorized officer

Marc S. Hoff

Telephone No. (703) 308-1677

Form PCT/ISA/210 (second sheet) (July 1998) \*

## INTERNATIONAL SEARCH REPORT

International application No.  
PCT/US01/04277

## C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 6,085,156 A (RUST et al.) 04 July 2000 (04.07.2000), see entire document.	1-5, 9-15, 18-20, 26-32, 36-51

Form PCT/ISA/210 (continuation of second sheet) (July 1998) ★

# INTERNATIONAL SEARCH REPORT

International application No.  
PCT/US01/04277

## B. FIELDS SEARCHED

Minimum documentation searched

Classification System: U.S.

702/57, 58, 60, 61, 62, 64, 65, 104, 108, 116, 119-126, 183-185, 188, 189, 103, 104, 106, 134, 135, 170, 171;  
700/9-12, 83, 84; 345/961, 970, 781; 709/213-229

## PATENT COOPERATION TREATY

## PCT

## INTERNATIONAL SEARCH REPORT

(PCT Article 18 and Rules 43 and 44)

Applicant's or agent's file reference 3165.1000002	<b>FOR FURTHER ACTION</b> see Form PCT/ISA/220 as well as, where applicable, item 5 below.	
International application No. PCT/IB2004/002797	International filing date (day/month/year) 26/08/2004	(Earliest) Priority Date (day/month/year) 09/10/2003
Applicant EINFALT EHF.		

This International Search Report has been prepared by this International Searching Authority and is transmitted to the applicant according to Article 18. A copy is being transmitted to the International Bureau.

This International Search Report consists of a total of 4 sheets.

☒ It is also accompanied by a copy of each prior art document cited in this report.

**1. Basis of the report**

- a. With regard to the **language**, the international search was carried out on the basis of the international application in the language in which it was filed, unless otherwise indicated under this item.

☐ The international search was carried out on the basis of a translation of the international application furnished to this Authority (Rule 23.1(b)).

- b. ☐ With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, see Box No. I.

2. ☐ **Certain claims were found unsearchable** (See Box II).

3. ☐ **Unity of invention is lacking** (see Box III).

4. With regard to the **title**,

☒ the text is approved as submitted by the applicant.

☐ the text has been established by this Authority to read as follows:

5. With regard to the **abstract**,

☒ the text is approved as submitted by the applicant.

☐ the text has been established, according to Rule 38.2(b), by this Authority as it appears in Box No. IV. The applicant may, within one month from the date of mailing of this international search report, submit comments to this Authority.

6. With regard to the **drawings**,

- a. the figure of the **drawings** to be published with the abstract is Figure No. \_\_\_\_\_

☐ as suggested by the applicant.

☐ as selected by this Authority, because the applicant failed to suggest a figure.

☐ as selected by this Authority, because this figure better characterizes the invention.

- b. ☒ none of the figures is to be published with the abstract.

**A. CLASSIFICATION OF SUBJECT MATTER**  
G06F9/44

According to International Patent Classification (IPC) or to both national classification and IPC

**B. FIELDS SEARCHED**Minimum documentation searched (classification system followed by classification symbols)  
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal

**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 01/59406 A (EMBEDDED LAB TECHNOLOGIES, LLC; MIN, KYUNG, YANG; PASSE, PAUL, J) 16 August 2001 (2001-08-16) page 1, line 23 - page 2, line 3 page 4, line 15 - line 20 page 12, line 23 - page 15, line 5 page 17, line 5 - line 20 page 19, line 5 - page 23, line 18 figures 3,4,7	1-23
X	US 6 286 017 B1 (EGILSSON Á ET AL) 4 September 2001 (2001-09-04) abstract column 2, line 29 - column 5, line 55 column 10, line 50 - column 12, line 6 figures 5-8,13,14 ----- -/--	24-54, 73-87

☒ Further documents are listed in the continuation of box C.☒ Patent family members are listed in annex.

## \* Special categories of cited documents:

- \*A\* document defining the general state of the art which is not considered to be of particular relevance
- \*E\* earlier document but published on or after the international filing date
- \*L\* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- \*O\* document referring to an oral disclosure, use, exhibition or other means
- \*P\* document published prior to the international filing date but later than the priority date claimed

- \*T\* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
- \*X\* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
- \*Y\* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
- \*G\* document member of the same patent family

Date of the actual completion of the international search

22 December 2005

Date of mailing of the international search report

10/01/2006

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentlaan 2  
NL - 2280 HV Rijswijk  
Tel. (+31-70) 340-2040, Tx. 31 651 epo nl,  
Fax: (+31-70) 340-3016

Authorized officer

Knapczyk, F

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT		
Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	<p>P. GÄNG, A. KAMENZ, H. VONHOEGEN:  "AUTOMATISATION AVEC DES MACROS".  LE GRAND LIVRE DE EXCEL 5, 1994, pages  851-858, XP002360093  PARIS  ISBN: 2-7429-0163-9  page 851, line 11 - line 22  page 852, line 29 - page 853, line 10  figure 6</p>	55-72, 88-97
A	<p>-----</p> <p>UNKNOWN: "FOR (MACRO SHEETS ONLY)"  EXCEL MACRO HELP FILE, 'Online!  31 October 2001 (2001-10-31), pages 1-1,  XP002360094  INTERNET  Retrieved from the Internet:  URL: <a href="http://web.archive.org/web/20011212233743/www.xl-logic.com/xl_files/addins/macro_fun.zip">http://web.archive.org/web/20011212233743/www.xl-logic.com/xl_files/addins/macro_fun.zip</a>  &gt; 'retrieved on 2005-12-16!  the whole document</p>	88-97
A	<p>-----</p> <p>US 5 680 557 A (KARAMCHETTY ET AL)  21 October 1997 (1997-10-21)  abstract  column 21, line 40 - column 23, line 29  figures 71-77</p> <p>-----</p>	24-54, 73-87

## INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No

PCT/IB2004/002797

Patent document cited in search report		Publication date	Patent family member(s)		Publication date
WO 0159406	A	16-08-2001	AU	3811301 A	20-08-2001
US 6286017	B1	04-09-2001	AU	4783796 A	11-09-1996
			CA	2214972 A1	29-08-1996
			DE	69600794 D1	19-11-1998
			DE	69600794 T2	02-06-1999
			WO	9626484 A2	29-08-1996
			EP	0811193 A2	10-12-1997
US 5680557	A	21-10-1997	NONE		

**This Page is Inserted by IFW Indexing and Scanning  
Operations and is not part of the Official Record**

**BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☒ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☒ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** \_\_\_\_\_

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.**